UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE CIENCIAS

DEPARTAMENTO ACADEMICO DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE INVESTIGACIÓN

TESIS PARA OBTENER EL TITULO PROFESIONAL DE INGENIERO ELECTRONICO Y TELECOMUNICACIONES

TITULO DEL PROYECTO:

SISTEMA DE CONTROL DE SILLA DE RUEDAS PARA PERSONAS PARAPLÉJICAS Y TETRAPLÉJICAS, USANDO MOVIMIENTO TRASLACIONAL, VOZ, BLUETOOTH Y S.O. ANDROID.

PERSONAL INVESTIGADOR:

BACH. JUNIOR ISRAEL ORDINOLA ROMAN.

BACH. WILSON TICLIAHUANCA TIMOTEO.

DOCENTE ASESOR:

ING. FRANKLIN BARRA ZAPATA

PIURA – PERU

2014

TITULO DEL PROYECTO

"SISTEMA DE CONTROL DE SILLA DE RUEDAS PARA PERSONAS PARAPLÉJICAS Y TETRAPLÉJICAS, USANDO MOVIMIENTO TRASLACIONAL, VOZ, BLUETOOTH Y S.O. ANDROID"

RESPONSABLES DEL DESARROLLO DE LA TESIS:

BACH. ORDINOLA ROMAN JUNIOR ISRAEL

BACH. TICLIAHUANCA TIMOTEO WILSON

ASESOR: ING. FRANKLIN BARRA ZAPATA

JURADO EVALUADOR:

PRESIDENTE: ING EDWIN ARNALDO OCAS INFANTE

SECRETARIO: ING EDUARDO OMAR AVILA REGALADO

VOCAL: ING CARLOS ENRIQUE ARELLANO RAMIREZ

DEDICATORIA

A Dios, por habernos permitido llegar hasta este punto y habernos dado la vida para lograr nuestros objetivos, además de su infinita bondad y amor.

A nuestros padres por su incondicional apoyo "Sin ustedes que difícil hubiera sido llegar al final de este largo camino".

AGRADECIMIENTO

A Dios, quien nos dio la confianza, sabiduría y fortaleza necesarias para finalizar de manera exitosa la realización de este trabajo; a nuestros padres y hermanos por el apoyo que nos brindaron a lo largo del desarrollo de esta tesis; al Ing. Franklin Barra Zapata que nos brindó toda su ayuda para la finalización de este trabajo

INDICE GENERAL

INDICE GENERAL

Pág

CAPITULO I

CAPITULO II

CAPITULO III

CAPITULO IV

INDICE DE TABLAS

INDICE DE FIGURAS

CAPITULO I

1. MARCO TEORICO

1.1. CARACTERÍSTICAS DE LA FAMILIA ATMEL

La familia de Microcontroladores AVR, pertenecen a ATMEL, los cuales poseen una arquitectura moderna. Estos Microcontroladores están divididos en tres grupos: TinyAVR, AVR Clásico y MegaAVR. Se muestran en la tabla 1.1 los dispositivos Microcontroladores de la serie AVR. Todos ellos se fabrican en el mismo proceso y los mismos niveles de implantación. Los dispositivos varían en densidad de memoria.

Con 1K byte Flash	Con 2K byte Flash	Con 4K byte Flash	Con 8K byte Flash	Con 12K byte Flash	Con 16K byte Flash	Con 32K/40K byte Flash	Con 64K byte Flash	Con 128K byte Flash	Con 256K byte Flash
Tiny13	Tiny14	Mega48	Mega8	90VC8544	Mega16	Mega32	Mega64	Mega128	Mega2560
	Tiny25	Tiny45	Mega8515		Mega162	Mega325	Mega645	Mega1280	Mega2561
	Tiny26		Mega8535		Mega169	Mega329	Mega649	Mega1281	
	Tiny2313		Mega88		Mega165	Mega406	Mega644		
					Mega165				

Tabla 1.1: Microcontroladores ATMEL

1.1.1 ARQUITECTURA DE LOS MICROCONTROLADORES AVR ATMEGA32

1.1.1.1 NÚCLEO DEL CPU

La principal función del núcleo del CPU es asegurar la correcta ejecución de un programa. El CPU debe ser capaz de acceder a la memoria, ejecutar cálculos, controlar los periféricos y manejar las interrupciones. Se muestra el diagrama a bloques de un núcleo CPU.

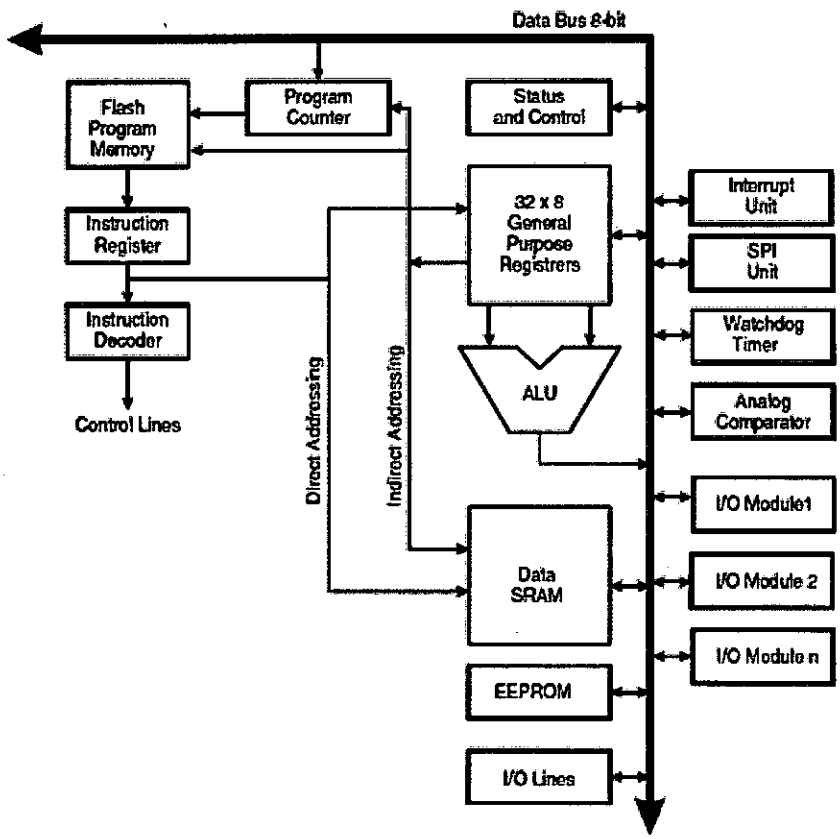


Fig. 1.1 Núcleo del CPU

1.1.2 REGISTRO

Es un espacio especial de almacenamiento de 8 bits, el cual luce como:

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

Note que la numeración de estos bits: el bit menos significativo inicia con cero (2^0 =1). Un registro puede almacenar números desde 0 a 255 (números positivos), o números desde -128 a 127 (números con signo y magnitud), o valores que representan caracteres en el código ASCII.

Las características especiales de los registros, comparados con otros sitios de almacenamiento, es que:

- Ellos se usan directamente en los comandos en ensamblador.
- Las operaciones de su contenido requieren solamente comandos de una palabra.
- Ellos están ligados directamente a la Unidad Central de Procesamiento.
- Ellos son la fuente o destino de los argumentos en las operaciones.

Existen 32 registros en un AVR. Ellos se nombran como R0 a R31, pero puedes renombrarlos usando las directivas del ensamblador.
Los AVR utilizan la Arquitectura Harvard, con el bus de datos y el bus de memorias separados. Mientras una instrucción se ejecuta, la próxima instrucción esta lista para ser ejecutada en la memoria de programa. Las instrucciones se ejecutan en cada ciclo de reloj. La memoria de programa está en la memoria Flash. Al ejecutarse una operación en la ALU, los dos operandos están a la salida del archivo de registros y el resultado se almacena al fondo del archivo de registros en un solo ciclo de reloj. Seis de los 32 registros se pueden usar como registros apuntadores de direccionamiento indirecto a 16 bits para datos almacenados en memoria, siendo los registros de 16 bits X, Y y Z. Después de realizar una operación aritmética el Registro de Estado actualiza la información acerca del resultado de la operación.

Durante las interrupciones o llamados a subrutinas, la dirección del Contador de Programa se almacena en la pila. La pila se localiza en la SRAM. El usuario al inicio de un programa deberá inicializar el SP en la rutina de reset.

1.1.3 REGISTRO DE ESTADO

Este registro contiene información del resultado más reciente de la operación aritmética. Así, esta información puede ser útil para alterar el flujo del programa y ejecutar operaciones condicionales. Puede ser manejado por software. Se muestra los bits que maneja.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 1.2 Registro Aritmético

1.1.4 REGISTROS DE TRABAJO DE PROPOSITO GENERAL

A continuación se observa el archivo de registros, donde los 32 registros de propósito general de 8 bits se localizan al inicio de las direcciones de memoria de la SRAM (0000h a la 001Fh).

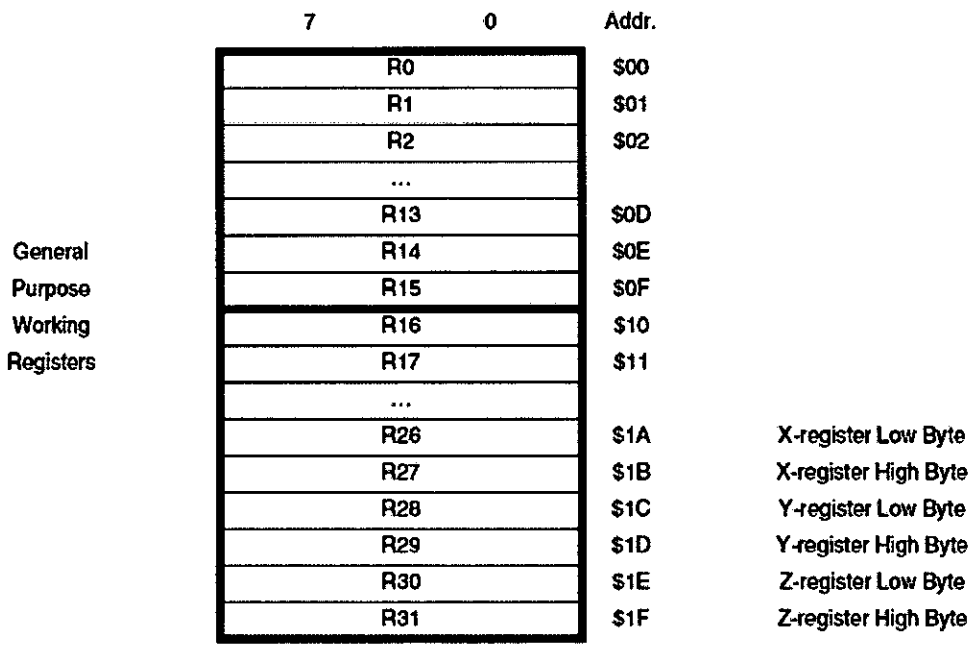


Fig. 1.3 Archivo de registros

A partir de la dirección 001Ah a la 001Fh, comienzan los 3 registros de 16 bits (X, Y y Z), los cuales actúan como apuntadores de direcciones para direccionamiento indirecto a espacio de datos.

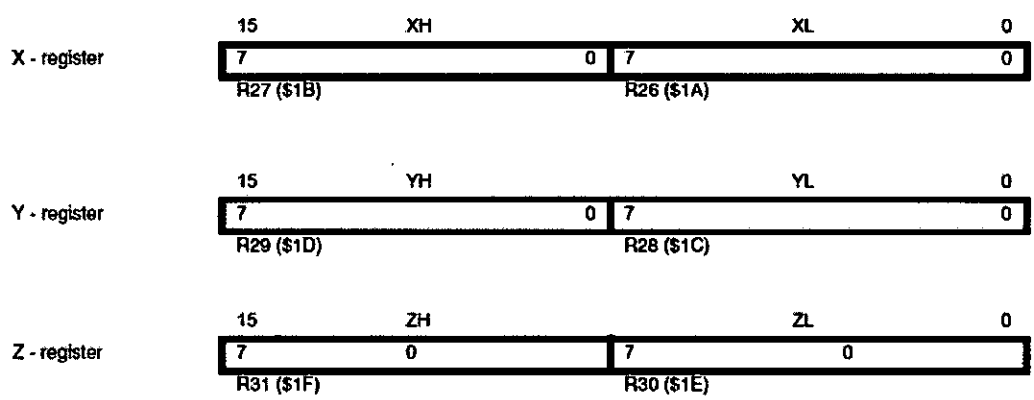


Fig. 1.4 Registro de 16 bits

1.2 MEMORIA DE PROGRAMA Y DE DATOS DEL AVR ATmega32

La arquitectura AVR tiene dos espacios principales de memoria, la Memoria de Datos y la Memoria de Programa. Así como también el Atmega32 posee la Memoria EEPROM para almacenar datos.

El Atmega32 contiene 32K bytes de memoria Flash Reprogramable para almacenar el programa. Ya que todas las instrucciones de los AVR son de 16 o 32 bits de ancho, la Flash se organiza en 16K x 16. La memoria de Programa Flash se divide en dos secciones, la sección de "Booteo" de Programa y la sección de Aplicación del Programa. El contador de programa (PC) es de 14 bits de ancho, así direcciona localidades en la memoria de programa de 16K.

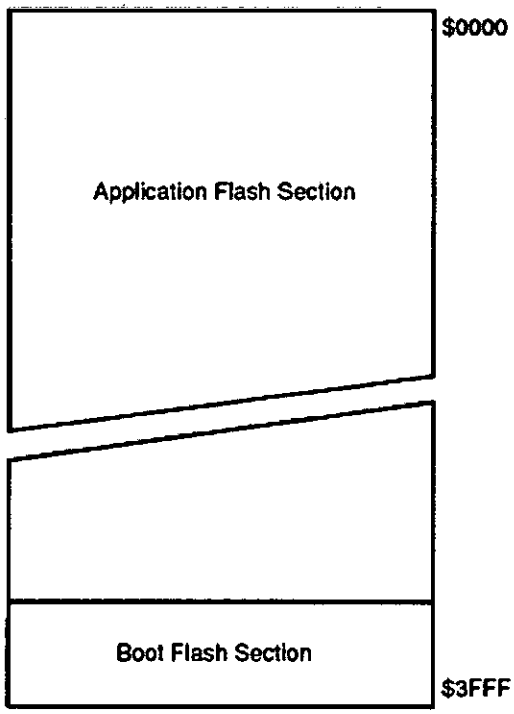


Fig. 1.5 Localidades de Memoria

Las 2144 (860h) localidades de Memoria de Datos direccionan el Archivo de Registros, la Memoria de I/O y los datos internos de la SRAM. Las primeras 96 (60h) localidades direccionan el Archivo de Registros y la Memoria de I/O, y las siguientes 2048 localidades direccionan los datos internos de la SRAM, ver la siguiente figura.

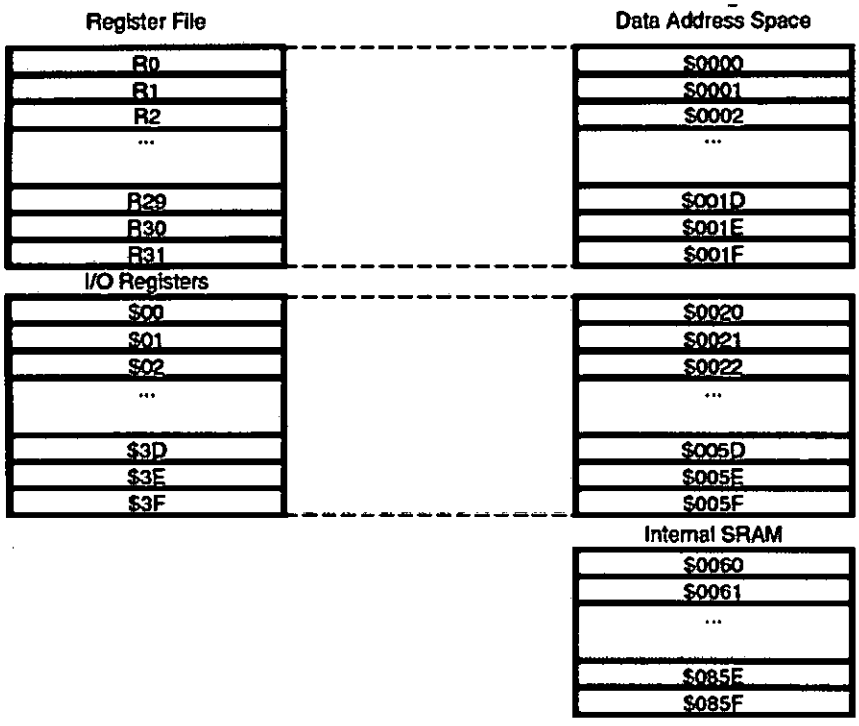


Fig. 1.6 Archivo de Registros y la Memoria de I/O

1.3 CARACTERÍSTICAS DE LOS MICROCONTROLADORES ATmega32.

ATmega32 es un microcontrolador CMOS de 8 bits a baja potencia basado en arquitectura RISC de AVR. Ejecutando las instrucciones en un solo ciclo de reloj, el ATmega32 alcanza un desempeño de 1 MIPS por MHz permitiendo al diseñador optimizar consumos de potencia contra la velocidad de procesamiento. Las características generales del ATmega32 son:

- ATmega32 (Serie AVR de Atmel de 8 bits).
- Arquitectura RISC
- 32K bytes de memoria flash, 2K bytes de SRAM, 1024 bytes EEPROM, 2
- Timers/Contadores de 8 bits, 1 Timer/Contador de 16 bits, 8 canales de 10 bits de ADC, USART, WDT, POR, BOD, 4 Canales de PWM, Puerto de ISP.
- Interface Serial SPI para programación dentro del sistema.
- 6 Modos para ahorrar potencia.
- 32 pines de I/O.

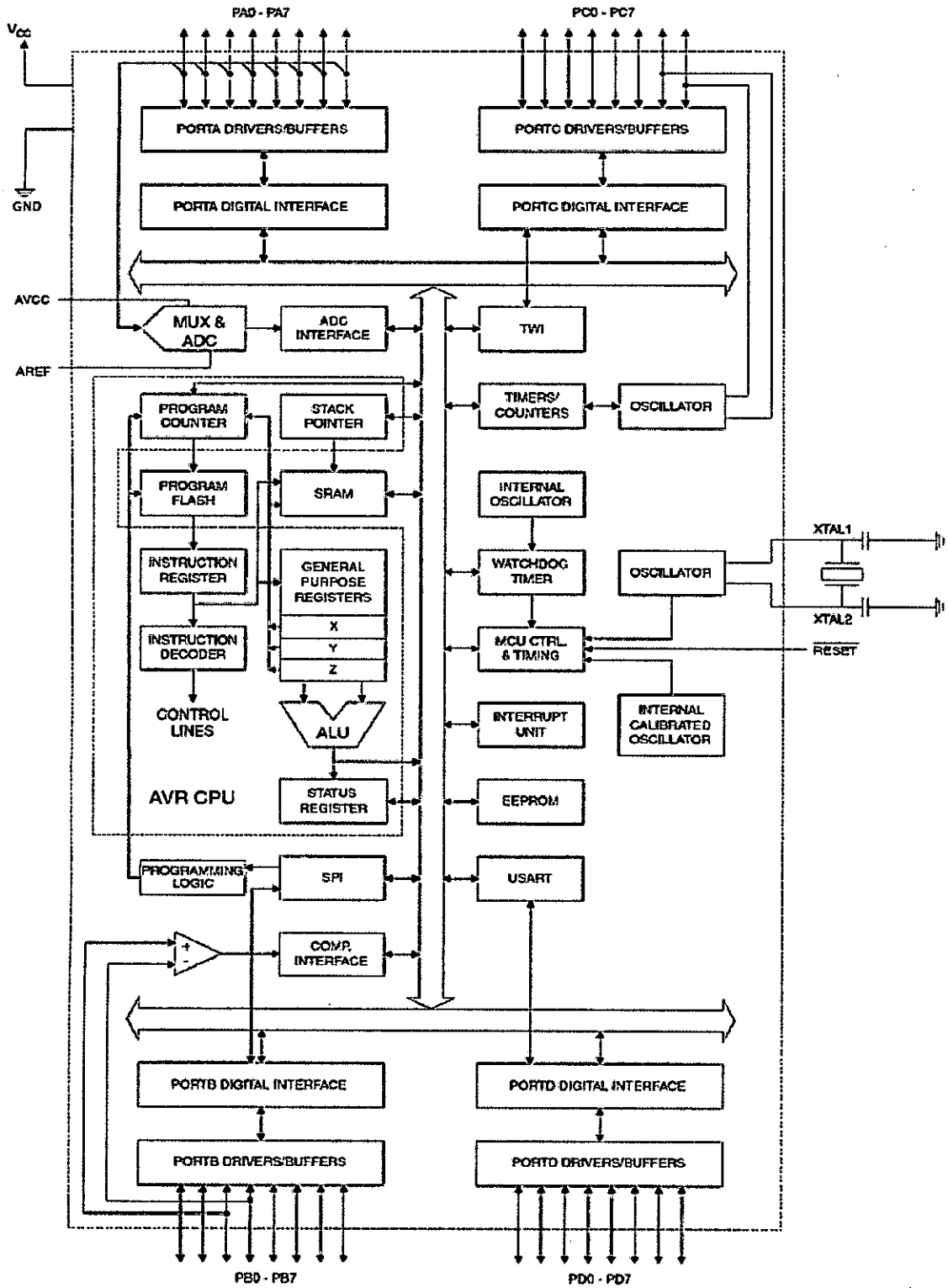


Fig. 1.7 Diagrama de bloques del ATmega32

El núcleo AVR posee un conjunto de instrucciones con 32 registros de trabajo de propósito general. Los 32 registros se conectan directamente a la Unidad Aritmética y Lógica (ALU), permitiendo a dos registros solamente acceder en una sola instrucción y sean ejecutadas en sólo un ciclo de reloj. Alcanzando un desempeño de 10 veces más rápido que los microcontroladores con tecnología CISC. El ATmega32 tiene las características: 32K bytes de memoria Flash programable dentro del sistema, 1024 bytes de EEPROM, 2K bytes de SRAM, 32 líneas de I/O de propósito general, 32 registros de propósito general, Interfase JTAG, 3 Timers/Contadores con modos de comparación, interrupciones internas y externas, un USART programable, una interfase serial orientada a byte de dos líneas, 8 canales de convertidor Analógico- Digital de 10 bits, con opción a ser diferenciales, un timer perro guardian (watchdog) con oscilador interno que es un temporizador especial que se puede activar en cualquier sección del código y cuando está activado se asegura de que un cierto número de instrucciones se ejecutan en un tiempo predefinido, un puerto serial SPI, y 6 modos de ahorrar potencia.

El convertidor Analógico Digital del ATmega32 es por Aproximaciones Sucesivas con una resolución de 10 bits. El ADC se conecta a un multiplexor de 8 canales análogos el cual permite 8 voltajes de entrada en una sola terminación construido en los pines del puerto A. El dispositivo también soporta 16 combinaciones de voltajes de entrada diferenciales. Dos de las entradas diferenciales (ADC1, ADC0 y ADC3, ADC2) están equipadas con una etapa de ganancia programable, proveyendo pasos de amplificación de 0dB (1x), 20dB(10x), o 46dB (200x) en el voltaje de entrada diferencial antes de la conversión del A/D.

El modo de ahorrar potencia salva el contenido de los registros pero congela al oscilador, deshabilitando todas las funciones de CI hasta la próxima interrupción o reinicio del Hardware. En el modo de ahorrar potencia, el timer asíncrono continua corriendo, permitiendo al usuario mantener un tiempo base mientras el resto del dispositivo esta "durmiendo". Esto permite un ahorro de potencia.
El ATmega32 AVR soporta: compiladores en C, macro ensambladores, simuladores en debugger, emuladores dentro del circuito y los kits de evaluación.

1.4. TERMINALES DE LOS MICROCONTROLADORES ATmega32.

A continuación se detallan las terminales del ATmega32.

Vcc: Fuente de voltaje digital (5 Volts)

GND: Tierra.

Puerto A (PA7..PA0)

El puerto A sirve como entradas analógicas al convertidor ADC. Además el puerto A sirve como puerto de 8 bits de I/O bidireccionales, si el Convertidor A/D no es usado. El buffer de salida del puerto A tiene la capacidad de abastecer y drenar corriente.

Cuando los pines PA0 a PA7 son usados como entrada y externamente jalados hacia abajo, ellos abastecen corriente si los resistores internos pull-up s activan. Los pines del puerto A son de tres estados cuando la condición de reset se activa.

Puerto B (PB7..PB0)

El puerto B es un puerto de 8 bits de I/O bidireccionales con resistores internos pull-up (seleccionados por cada bit). El buffer de salida del puerto B tiene la capacidad de abastecer y drenar corriente. Cuando los pines PB0 a PB7 son usados como entrada y externamente jalados hacia abajo, ellos abastecen corriente si los resistores internos pull-up se activan. Los pines del puerto B son de tres estados cuando la condición de reset se activa. El puerto B también alberga funciones de registros de especiales, como se enlistan en la Tabla.

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

Tabla 1.2 Puerto B del ATmega32

Puerto C (PC7..PC0)

El puerto C es un puerto de 8 bits de I/O bidireccionales con resistores internos pull-up (seleccionados por cada bit). El buffer de salida del puerto C tiene la capacidad de abastecer y drenar corriente. Cuando los pines PC0 a PC7 son usados como entrada y externamente jalados hacia abajo, ellos abastecen corriente si los resistores internos pull-up s activan. Los pines del puerto C son de tres estados cuando la condición de reset se activa.

Puerto D (PD7..PD0)

El puerto D es un puerto de 8 bits de I/O bidireccionales con resistores internos pull-up (seleccionados por cada bit). El buffer de salida del puerto D tiene la capacidad de abastecer y drenar corriente. Cuando los pines PD0 a PD7 son usados como entrada y externamente jalados hacia abajo, ellos abastecen corriente si los resistores internos pull-up s activan. Los pines del puerto D son de tres estados cuando la condición de reset se activa. El puerto D también alberga funciones de registros de especiales, como se enlistan en la Tabla.

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

Tabla 1.3 Puerto D del ATmega32

RESET: Entrada de reinicio (RESET). Un nivel bajo en este pin aplicado por mas de un tiempo del mínimo pulso (1.5 trst = µs) generara un reset, aún si el reloj no esta corriendo.
XTAL1: Entrada del amplificador inversor que forma parte del oscilador.
XTAL2: Salida del amplificador inversor que forma parte del oscilador. Como se muestra en la Fig.

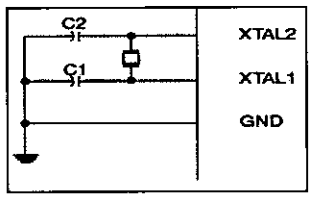


Fig. 1.8 Conexión del XTAL en el ATmega32

AVCC: Es el pin de la fuente de voltaje para el Puerto A del Convertidor A/D. deberá ser conectada a Vcc, aún si el ADC no se utiliza. Si el ADC se utiliza se conecta a Vcc a través de un filtro pasa bajo.

AREF: Es el pin de referencia analógica para el convertidor A/D. La fig. muestra la distribución de terminales del microcontrolador para un encapsulado PDIP,

WATCHDOG TIMER El watchdog timer es un temporizador de 16 bit.

Al iniciarse un programa, el watchdog timer está activo y configurado por defecto con un intervalo de reset de ~32 [ms]. La principal función del watchdog timer (WDT) es reiniciar el procesador después de que ocurra una falla o problema de software, o después de un intervalo de tiempo determinado generado por el programador, en cuyo caso se reinicia el procesador o el programa en ejecución.

Si el watchdog timer no se emplea en ninguna subrutina puede ser configurado como un temporizador de intervalos y puede generar interrupciones en los intervalos de tiempo seleccionados

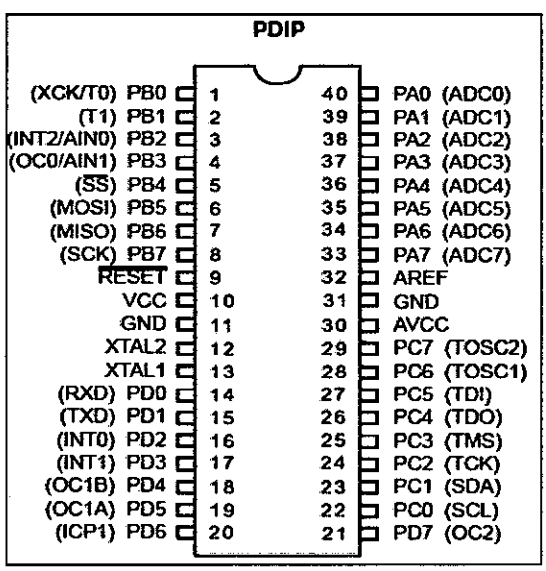


Fig. 1.9 Distribución de terminales del ATmega32, empaquetado PDIP

1.5. CONECTAR UN MOSFET DE POTENCIA A UN MICROCONTROLADOR

Se describirá en modo simple y ejemplificado, la conexión de Mosfets a microcontroladores u otros circuitos digitales para controlar motores, leds o cualquier dispositivo de potencia que trabaje con baja tensión continua (DC).

Los Mosfets de potencia (Power Mosfets) son componentes electrónicos que nos permiten controlar corrientes muy elevadas. Como en el caso del los Mosfets comunes, tienen tres terminales de salida que se llaman: Drain, Source y Gate (D, S y G). La corriente principal pasa entre Source y Drain (I_{SD}) mientras que el control de esta corriente se obtiene aplicando una tensión sobre el terminal Gate (respecto al terminal Source), conocida como V_{GS}.

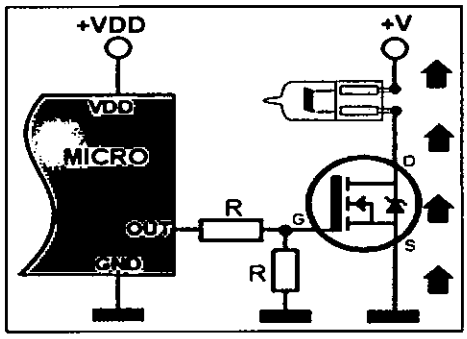


Fig. 1.10 Conexión directa de un Mosfet a un microcontrolador

En condiciones de reposo, la corriente de Gate es prácticamente nula porque al interno del componente, el terminal Gate se encuentra conectado a una especie de capacitor. Por lo tanto circula corriente de Gate solo cuando cambiamos el nivel de tensión de entrada (cambio de estado lógico) y este es el motivo por el cual el consumo de los Mosfet (como en el caso de todos los circuitos lógicos MOS) aumenta en proporción a la frecuencia de conmutación.

Existen "Power Mosfets" de dos tipos: los de canal N y los de canal P. La diferencia entre estos está en la polaridad de conexión Source-Drain y en el hecho que la tensión de Gate de los Mosfet de canal P es negativa (las mismas diferencias que existen entre los transistores NPN y PNP).

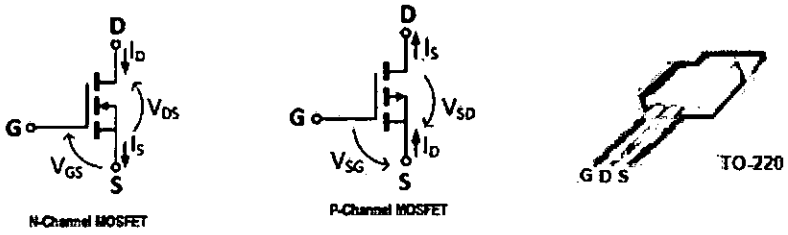


Fig. 1.11 Símbolos de los Mosfets

En base a la aplicación, un Mosfet de potencia puede trabajar en la "región lineal" o en "saturación". En los sistemas analógicos, como por ejemplo en las etapas de salida de los amplificadores de audio, los Mosfets trabajan en la región lineal mientras que en los sistemas digitales, en los cuales se usan como interruptores digitales de potencia, estos trabajan en corte (OFF) o saturación (ON). En este artículo analizaremos solamente los Mosfet usados como interruptores digitales.

Cuando un Mosfet se encuentra en saturación, el valor de resistencia interno entre Source y Drain (Rsd) es muy bajo y por lo tanto, la disipación de potencia en él será poco significativa no obstante la corriente que lo atraviesa pueda ser muy elevada.

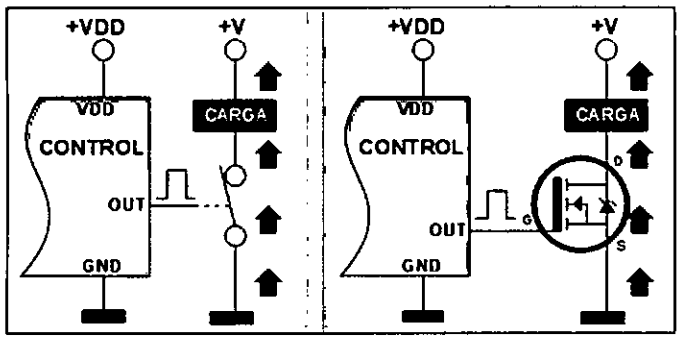


Fig. 1.12 Mosfet usado como simple interruptor de potencia

Para llevar un Mosfet a la saturación, es necesario que la tensión de control en el terminal Gate sea suficientemente alta y esto podría ser un problema si usáramos directamente la baja tensión de salida de un microcontrolador. Me explico mejor con un ejemplo. Para saturar un transistor bipolar (tipo BC548) se necesita superar la tensión de umbral de la base que es solamente de 0,6V. Una tensión de control de 0,6V se pueden obtener con cualquier sistema digital alimentado con 5V, 3,3V y hasta 1,8V. Por el contrario, la tensión necesaria para poner en conducción un Mosfet (llamada "tensión de umbral" o V_{th}) es

mucho más elevada (algunos volts) y depende del modelo de Mosfet. Es más, aunque si alcanzáramos este valor, no sería suficiente porque deberíamos salir de la región lineal de trabajo para llevarlo a la saturación. Si no fuera así, la conducción no sería plena y por lo tanto parte de la potencia se disiparía en el Mosfet en forma de calor porque la potencia disipada por el Mosfet es el resultado de la multiplicación entre la caída de tensión y la corriente que pasa por él (PMosfet = Vsd * Isd).

En la siguiente figura, se muestran las curvas de entrada y salida de un transistor MOSFET N con V_t= 2V conectado en Fuente común (SC), es decir, el terminal de Fuente, es común la señal de entrada V_{GS} y las señales de salida I_D y V_{DS}.

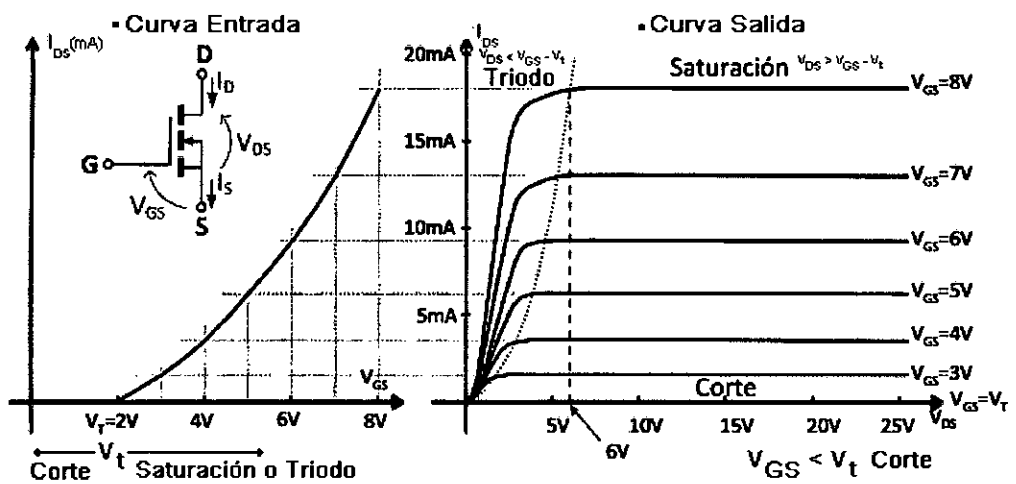


Fig. 1.13 Curvas de conducción de un hipotético Mosfet con las dos regiones de trabajo (lineal y saturación) para distintas tensiones de Gate

Estas curvas de salida, se obtienen al representar las variaciones de I_D al aumentar V_{DS}, para diferentes valores de V_{GS}, es decir, $I_D = f(Vds)_{VGS=cte}$.

La curva más baja es la curva de $V_{GS(T)}$. Cuando V_{GS} es menor que $V_{GS(T)}$, la corriente de Drenador es extremadamente pequeña. Cuando V_{GS} es mayor que $V_{GS(T)}$, fluye una considerable corriente, cuyo valor depende de V_{GS}.

Si $V_{GS} \leq V_T$, el transistor MOSFET, estará en la región de *corte* y la corriente I_D=0.

Si $V_{GS} \geq V_T$, el transistor MOSFET, estará en la *región de conducción* y se pueden dar dos casos:

a) Si $V_{DS} \geq V_{GS}-V_T$, el transistor MOSFET, estará en la *región de saturación* y la corriente será constante para un valor determinado de V_{GS}. La curva de transferencia de la figura que representa $I_D=f(V_{DS})_{VGS=cte.}$, se obtiene a partir de las curvas de salida

para una tensión V_{DS} constante que sitúe al transistor en saturación. Se observa que aproximadamente corresponde a la curva de una parábola con vértice en V_T y por tanto, la corriente puede determinarse de forma aproximada por:

$$I_D = k(V_{GS}-V_T)^2$$

Donde k es el parámetro de transconductancia del MOSFET N y se mide en mA/V^2.

b) Si $V_{DS} \leq V_{GS}-V_T$, el transistor MOSFET, estará en la región óhmica de forma que, al aumentar V_{DS}, también lo harán la corriente y la resistencia del canal. El comportamiento del transistor puede asociarse a la resistencia que presenta el canal entre Drenador y Fuente.

Los Mosfets con baja tensión de Gate son conocidos con el nombre de "logic level power Mosfet" (Mosfet de potencia para nivel lógico).

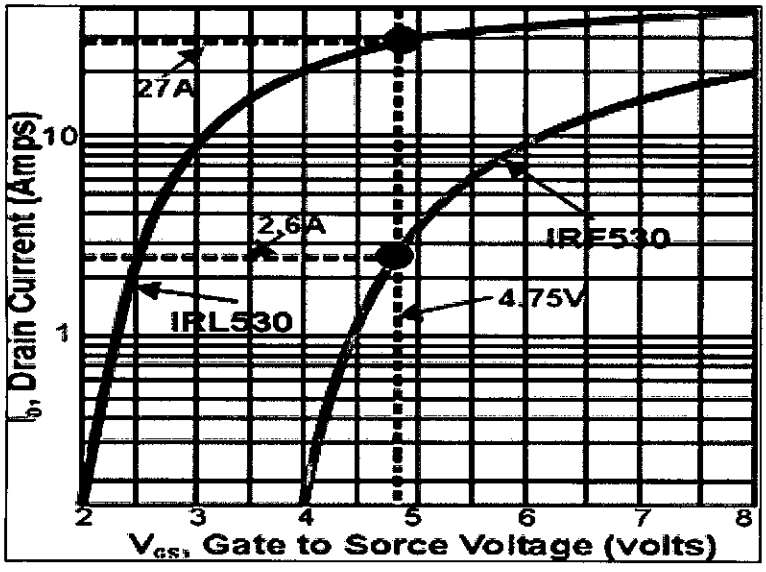


Fig. 1.14 Comparación entre un Mosfet "Logic level" y un Mosfet común

En el diseño podemos ver la curva de conducción de un Mosfet "logic level" IRL530 (en verde) comparada con un clásico Mosfet IRF530 (en azul). La línea vertical a rayas indica un nivel lógico de 4,75V (típico nivel de salida de un microcontrolador alimentado con 5V). Como podemos observar, la corriente de salida máxima con el IRF530 no supera los 2,6A no obstante este modelo sea en grado de entregar mucha más corriente mientras que el IRL530 supera ampliamente los 20A (plena conducción). Si nuestro microcontrolador trabajara con 3,3V el IRF530 no lograría ni siquiera entrar en conducción.

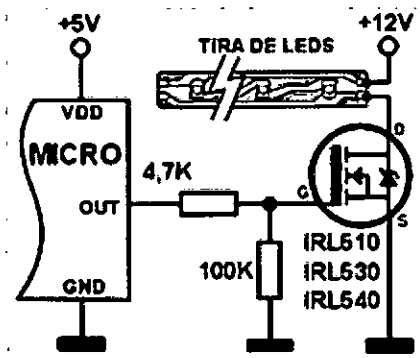


Fig. 1.15 Conexión de microcontrolador con tira led.

Por lo tanto, elegir un Mosfet de tipo "logic level" es la mejor elección cuando trabajamos con circuitos digitales. En la figura podemos observar la conexión de un Mosfet "logic level" a un microcontrolador para encender una tira de leds.

Como explicado al principio de este artículo, cuando cambia el nivel lógico de control, por un instante el Mosfet absorbe una cierta corriente que carga el capacitor interno del terminal Gate. La resistencia de 4,7K sirve para limitar esta corriente inicial. Podríamos usar cualquier valor de resistencia pero un valor bajo permite la carga rápida de este capacitor y por lo tanto una conmutación más veloz del Mosfet, útil si quisiéramos usar una regulación de potencia por impulsos (PWM). En este tipo de regulación, si la conmutación del Mosfet fuera "lenta", este se encontraría por más tiempo en la zona lineal y por lo tanto aumentaría la disipación de potencia en él, especialmente si trabajamos con frecuencias elevadas. Una vez que el Mosfet ha conmutado, el Gate no absorbe más corriente. Por lo tanto si pensamos de usar nuestro Mosfet para simples encendidos y apagados, el valor de esta R puede ser de 10K. Por el contrario, si deseamos modular la potencia de salida a través de la modulación PWM, nos conviene un valor de resistencia de 4,7K, 3,3K o 1,2K inclusive. La mejor elección depende fundamentalmente de la frecuencia PWM.

La resistencia de 100K a masa sirve para definir un estado lógico preciso en el caso que el micro no lo hiciese como por ejemplo en la fase de inicialización del mismo.

Si tuviéramos la necesidad de conectar un Mosfet no "logic level" a un circuito digital, podemos agregar un transistor que nos permita aumentar la tensión de control como podemos observar en la figura siguiente.

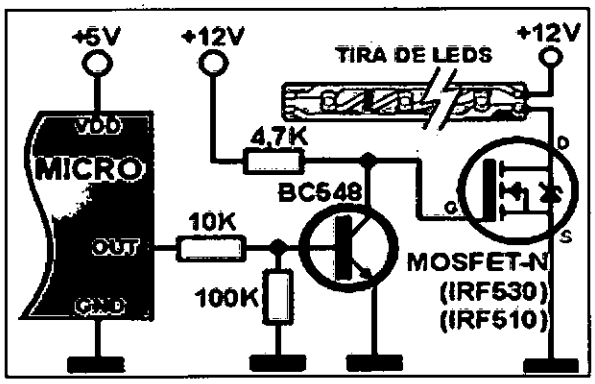


Fig. 1.16 Ejemplo de conexión de un Mosfet no "logic level" de canal N

El principio de funcionamiento es muy simple. Cuando la salida del microcontrolador tiene un nivel lógico bajo (0 volt), el transistor no conduce y por lo tanto, su colector, que se encuentra conectado al Gate del Mosfet tendrá un potencial positivo de 12V a través de la resistencia a positivo. Cuando la salida del microcontrolador pasa a nivel alto, (1,8V, 3,3V o 5V), el transistor conduce y lleva el Gate del Mosfet a 0V, por lo tanto el Mosfet deja de conducir. Como podrán observar, este circuito tiene el defecto que trabaja al contrario es decir, se activa cuando el nivel de salida del micro es bajo. No obstante esto, tiene la ventaja que la tensión de Gate alcanza la tensión máxima de alimentación garantizando la completa saturación de cualquier tipo de Mosfet que conectemos. El valor de la resistencia de gate conectada a positivo modifica la velocidad de conmutación del Mosfet como explicado en el caso anterior. (valores altos para conmutaciones lentas y valores bajos para conmutaciones veloces (modulación PWM).

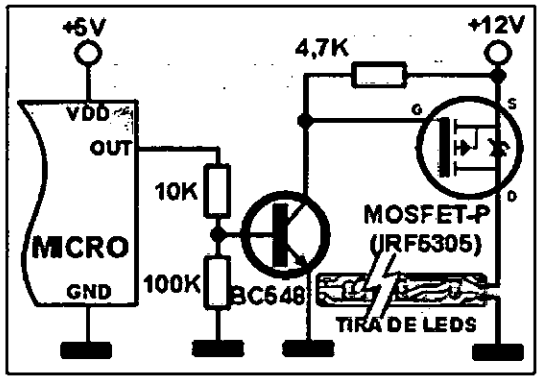


Fig. 1.17 Ejemplo de conexión de un Mosfet no "Logic Level" de canal P

Si quisiéramos usar un Mosfet común (no "logic level") con lógica de control no invertida, podemos cambiarlo por uno de canal P como se observa en la figura. Noten que la potencia de salida (en el ejemplo, la tira de leds) se conecta hacia masa (negativo) en lugar del positivo. El único problema que presenta esta última solución es no se puede usar si quisiéramos controlar una tira de leds RGB con 3 canales porque estas tiras, generalmente tienen el ánodo en común (positivo único) mientras que a nosotros nos serviría una tira RGB con cátodo común (negativo común). De cualquier manera, esta solución es muy útil en numerosos casos. Como en los otros ejemplos, el valor de la resistencia de gate conectada a positivo modifica la velocidad de conmutación del Mosfet (valores altos para conmutaciones lentas y valores bajos para conmutaciones veloces (modulación PWM).

1.6. ACELERÓMETRO

1.6.1. INTRODUCCIÓN

Los acelerómetros o sensores de aceleración, están pensados para realizar una medida de aceleración o vibración, proporcionando una señal eléctrica según la variación física, en este caso la variación física es la aceleración o la vibración.

Los rangos de medida son diversos, desde 1 g, hasta los miles de g´s. Respecto al rango de frecuencia disponible, hay acelerómetros que parten de 0 Hz, para medida de bajas frecuencias, acelerómetros que llegan hasta los miles de Hz para altas frecuencias de vibración, otros modelos de muy alta sensibilidad con bajo rango de frecuencia, etc.

Dispositivo	Rango de medida	interfaces	ejes	Ancho de banda	alimentacion	extras
ADXL335	±3g	Analógica	3	1600(x/y),550(z)Hz	1.8-3.6v, 350μA	Self test
MMA7361	±1.5g, ±6g	Analógica	3	400(x/y),300(z)Hz	2.2 V -3.6 V,3 μA	Self test

Tabla 1.4 Comparación de algunos acelerómetros en el mercado

1.5.2. ADXL335

Para detectar inclinación, o un ángulo un posible dispositivo sería una combinación de dos integrados denominada IMU (Inertial Measurament Unit), que dispone de un ADXL335 que es un acelerómetro de tres ejes, y un IDG-500 que es un giroscopio de dos ejes.

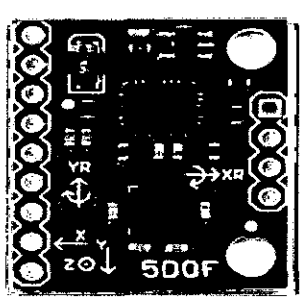


Fig. 1.18 IMU compuesta por el ADXL335 y el IDG-500

A continuación se explica el ADXL335, del que se detallan las características, el funcionamiento y los cálculos necesarios para obtener la inclinación.

El ADXL335 es un acelerómetro, y que como su nombre indica mide aceleración. Como sabemos la aceleración es la rapidez con la que un cuerpo gana o pierde velocidad, esta aceleración se puede medir en metros por segundo al cuadrado (m/s^2) o en fuerzas G que es una relación proporcional a la aceleración de la gravedad ($9'8m/s^2$), es decir que 1g = $9'8m/s^2$ y 2g = $19'6m/s^2$ que es el doble y así sucesivamente. Después de esta introducción, comentaremos las características más importantes para su elección y para finalizar explicaremos el funcionamiento interno.

1.6.2.1 PROPIEDADES

Las características a tener en cuenta son las siguientes:

- Rango de medidas
- Interface
- Número de ejes que mide
- Ancho de banda
- Alimentación
- Características extras

Ahora entraremos en profundidad en cada una de ellas.

- **Rango de medidas:** Este define el límite superior e inferior de medida del acelerómetro, según cuál sea el proyecto será importante definir bien cuáles son estos límites, por ejemplo en una aplicación para un móvil no es necesario coger un acelerómetro que mida hasta 250g, pero en una aplicación en un cohete espacial puede que incluso necesites más, otra cosa importante es adecuar bien los márgenes de seguridad y no coger unos muy grandes para asegurarnos que no saldremos del rango de medidas, ya que como más pequeño es el rango mayor sensibilidad tienes en la salida.
- **Interface:** En este apartado nos referiremos a como dan las acelerómetros la respuesta en su salida, en general hay de tres tipos: analógicos, PWM y digitales.

- Analógicos: Estos dispositivos producen un voltaje en la salida que es directamente proporcional a la aceleración que miden, normalmente cuando están en reposo (0g) proporcionan una señal que es la mitad del voltaje de alimentación. Esta interface es la más fácil de implementar, solo necesitas un convertidor analógico-digital y un microcontrolador.
- PWM: Con esta interface en la salida obtienes una señal cuadrada de frecuencia fija, pero con un duty cycle variable que depende de la aceleración que está midiendo.
- Digitales: Como dice su nombre estos dan una señal discretizada en su salida. Aunque estos son los que más opciones pueden tener y son los menos susceptibles al ruido externo, también son los más difíciles de implementar.

- **Número de ejes que mide:** Existen acelerómetros de 2 y 3 ejes, los de 2 ejes normalmente sirven para medir la aceleración en X e Y. Si se desea medir también aceleración en el eje Z es recomendable utilizar uno de 3 ejes.
- **Ancho de banda:** Es el rango en el cual el sensor pude medir con confianza la aceleración, esta puede variar mucho según su aplicación pero normalmente con un ancho de banda de 50-100Hz suele ser suficiente.
- **Alimentación:** Si el proyecto funciona con baterías, el saber cuánto consume el acelerómetro es importante para saber la autonomía que tendrá, si está conectado a la red esto ya no es tan importante aunque el consumo del proyecto se habría de minimizar al máximo.
- **Características extras:** Muchos de los nuevos acelerómetros incluyen opciones ingeniosas como rangos de medida seleccionables, detección de 0G, tipos de interface diferentes en el mismo integrado, etc. Una vez analizadas las características más importantes, ahora se verán las propiedades de este dispositivo

Fig. 1.19 Acelerómetro ADXL335

Dispositivo	Rango de medida	Interface	Ejes	Ancho de banda máximo	Alimentación	Extras
ADXL335	±3	Analógica	3	1600 (x/y), 550 (z) Hz	1.8-3.6 V, 350 µA	Self test

Tabla 1.5. Características ADXL335

Como podemos observar su rango de medidas no es muy elevado (±3g), pero analizando nuestro proyecto (telescopio), y viendo que su velocidad de movimiento es muy baja tampoco necesitábamos unos límites elevados de medidas. Como opción extra el dispositivo dispone de unas entradas para hacer un test de seguridad. Sabiendo que propiedades tiene nuestro dispositivo ya podemos entrar a ver cómo funciona y como nos puede servir para conocer la inclinación.

1.6.2.2. FUNCIONAMIENTO

Para entender mejor como funciona un acelerómetro, suele ser útil imaginarse el integrado como un cubo con una bola dentro. Si cogemos la caja con la bola y la ponemos en un sitio sin ningún campo gravitacional tendríamos lo que se ve a continuación.

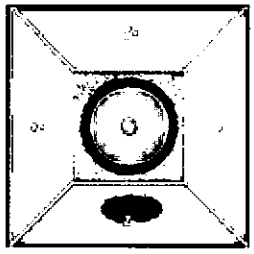


Fig. 1.20 Estado sin gravedad.

Si en este momento impulsas el cubo con una aceleración de 1g, hacia el eje X+, la bola golpeara con la pared X- con una fuerza de -1g

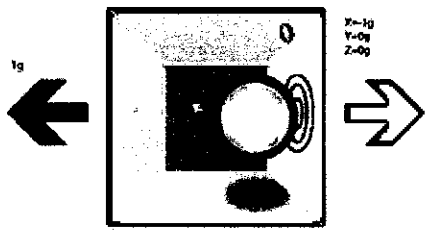


Fig. 1.21 Se le aplica al cubo una aceleración de 1g

Como se puede ver el acelerómetro detecta la fuerza directamente opuesta al vector aceleración esta fuerza opuesta es denominada fuerza de inercia. Ahora, analizaremos que pasa cuando se pone el acelerómetro en condiciones normales en las que hay una gravedad que le afecta, en primer lugar, al haber una gravedad que afecta el dispositivo la bola no estará en suspensión, como estaba antes, sino que se depositará en el suelo, como podemos ver en la siguiente figura.

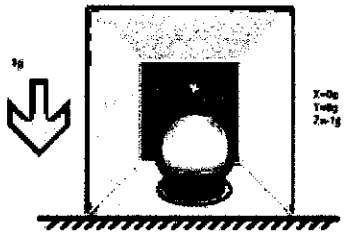


Fig. 1.22 Con el efecto del campo gravitatorio

En este caso la caja no se mueve pero ya se puede ver que en el eje Z- aparece una fuerza de -1g, esta fuerza es dada por la presión de la bola a causa de la gravedad. Como la gravedad le afecta siempre podemos medir cambios de orientación, por ejemplo si giramos el cubo 45 grados hacia la derecha obtenemos lo siguiente.

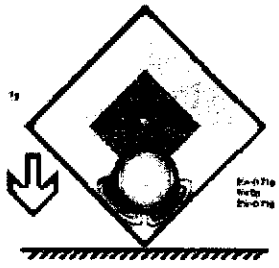


Fig. 1.23 Con un giro de 45 grados

Como podemos observar ahora la bola toca dos paredes lo que hace que la gravedad se reparta entre las dos superficies dando como resultado que en Z- hay una fuerza de -0.71 y en el X- una de también -0.71, este valor no arbitrario sino que es el resultado de la ecuación siguiente.

$$Z(-) = X(-) = 1g \cdot \cos(45) = \sqrt{\frac{1}{2}} \approx 0.71g$$

De esta forma es como un acelerómetro obtiene el valor de la fuerza que se le aplica.

1.7. SISTEMA OPERATIVO ANDROID

El sistema operativo Android fue creado originalmente para ser usado en dispositivos móviles de tercera generación pero luego se comenzó a utilizar en otros dispositivos móviles como notebooks, i-pods, mp3s, tablets y ahora se pueden ver hasta en electrodomésticos caseros como lavadoras y microondas.

En el año 2010, los teléfonos inteligentes con Android ocuparon el primer lugar en ventas en los Estados Unidos. En la actualidad, Android ostenta alrededor del 40% de cuota de mercado a escala mundial en lo que se refiere a teléfonos móviles de tercera generación situándose por delante de Symbian OS e iOS.

Una de las cosas que hacen de Android un sistema operativo para teléfonos móviles distinto a otros como el iOS y Windows Phone es que se desarrolla de forma abierta y se puede ingresar al código fuente así como al listado de incidencias, desde donde podemos ver problemas no resueltos y reportar problemas nuevos. En este artículo conoceremos más acerca de este sistema operativo.

Fig. 1.24 Logo Oficial de ANDROID

En el año 2005, Google compro la compañía Android Inc. Lo único que se sabía en ese momento era que Android desarrollaba un software para teléfonos móviles dando a entender que Google tenía en mente ingresar al mercado de la telefonía celular.
El 5 de diciembre del 2007, durante la inauguración de la "Open Handset Alliance" se estrenó Android como una plataforma de soporte para equipos móviles creados en la versión 2.6 del kernel de Linux. Desde entonces el avance de Android ha sido ascendente y hoy es uno de los sistemas operativos para móviles más usados del mundo.

1.7.1 CARACTERÍSTICAS:

Entre las principales características que posee Android podemos mencionar:

- **Conectividad:** Soporta tecnologías de conectividad como Wi-Fi, Bluetooth, GSM/EDGE, UMTS, WiMAX y otras más.
- **Mensajería:** Las formas más comunes como SMS y MMS están disponibles además del servicio PushMessaging de Android.
- **Video llamada:** Por medio de la versión HoneyComb, Android soporta video llamadas a través de Google Talk.
- **Soporte multimedia:** Puede soportar los formatos más conocidos como JPEG, MP3, MPEG-4, WAV, además de otros como WebM, H.263 y H.264.
- **Multi- táctil:** Android cuenta con soporte base para equipos móviles con pantallas multi-táctiles.
- **Almacenamiento:** Posee una base SQLite, la cual es utilizada para almacenamiento de datos.

PROGRAMACIÓN:

El desarrollo de aplicaciones para Android es sumamente sencillo y lo único que se necesita es un conocimiento básico de Java y poseer el kit de desarrollo de software provisto por Google. Este kit puede ser descargado completamente gratis.

DISPOSITIVOS:

Android es el sistema operativo con mayor presencia en dispositivos móviles como notebooks, tablets, i-pods, reproductores de mp3 y más. Android es uno de los sistemas operativos que ha logrado establecerse firmemente en el mercado en poco tiempo y además es el sistema con el mayor potencial de desarrollo en el mundo de la telefonía móvil.

Se calcula que en el actualidad hay más 400 000 aplicaciones para Android y que diariamente se activan alrededor de 500 000 equipos móviles.

1.8. PROGRAMACION EN APPINVENTOR

Google App Inventor fue una aplicación de Google Labs para crear aplicaciones de software para el sistema operativo Android. De forma visual ya partir de un conjunto de herramientas básicas, el usuario enlazando una serie de bloques para crear la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web. Las aplicaciones fruto de App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.

Con Google App Inventor, se espera un incremento importante en el número de aplicaciones para Android debido a dos grandes factores: la simplicidad de uso, que facilitará la aparición de un gran número de nuevas aplicaciones; y el Android Market , el centro de distribución de aplicaciones para Android donde cualquier usuario puede distribuir sus creaciones libremente.

1.8.1. HISTORIA

La aplicación se puso a disposición el 12 de julio de 2010 y está dirigida a personas que no están familiarizadas con la programación con ordenadores. En la creación de App

Inventor, Google se basó en investigaciones previas significativas en informática educativa. Fue creada a mediados del 2009 el profesor Harold Abelson del MIT.

Antes de salir al mercado se ha probado en diferentes centros educativos y la han utilizado desde niños de 12 años hasta licenciados universitarios sin nociones de programación.

A principio de agosto de 2011 Google anunció que ya no mantendría esta aplicación, pero que la haría código libre destinado a la educación.

Una semana después el **Instituto Tecnológico de Massachusetts (MIT),** una institución de educación superior privada situada en Cambridge, Massachusetts (EE.UU.), anunció que se haría cargo del proyecto.

El 31 de diciembre de 2011 App Inventor de Google dejó de funcionar. El 4 de marzo de 2012, el Instituto Tecnológico de Massachusetts (MIT) volvió a poner el proyecto en Internet.

1.8.2. CARACTERISTICAS

El editor de bloques de la aplicación utiliza la librería Open Blocks de Java para crear un lenguaje visual a partir de bloques. Estas librerías están distribuidas por Massachusetts Institute of Technology(MIT) bajo su licencia libre (MIT License). El compilador que traduce el lenguaje visual de los bloques para la aplicación en Android utiliza Kawa como lenguaje de programación, distribuido como parte del sistema operativo GNU de la Free Software Foundation

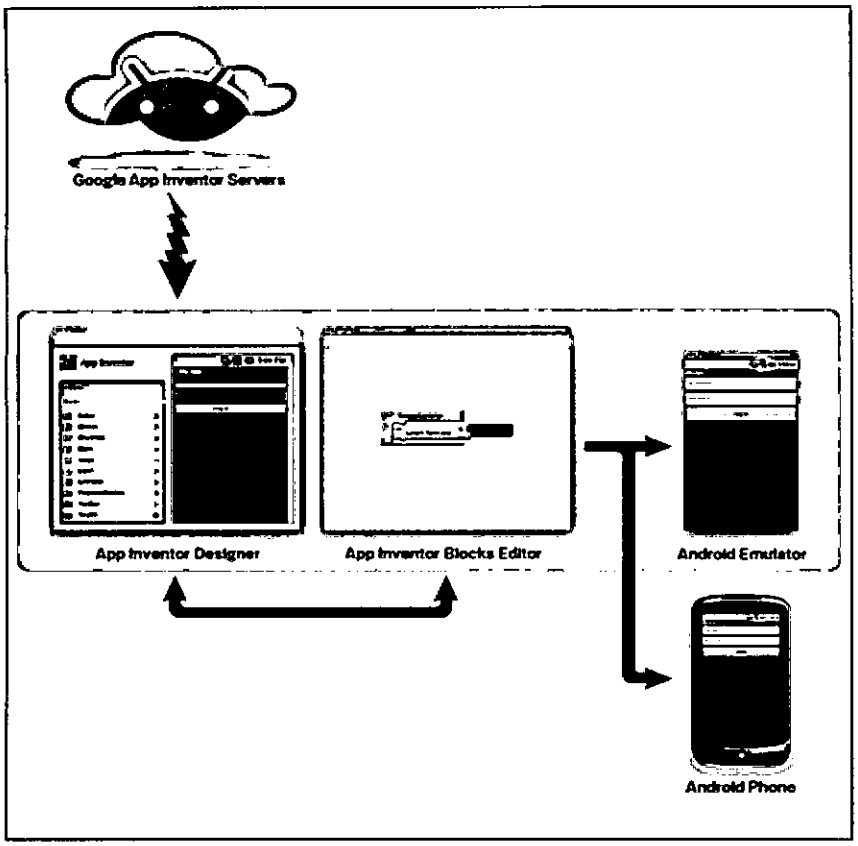


Fig. 1.25. Programando en APPINVENOR

El App inventor no es un programa que se baja al ordenador. Funciona en cloudcomputing, el programa funciona mediante la conexión a Internet, se trabaja con el programa conectado directamente al servidor de App inventor. Es como una página web y en ella realizas las operaciones.

Para que funcione se necesita tener instalado JAVA. También asociar el correo del usuario a Google, esto se realiza inmediatamente sin dificultad. Para crear una aplicación con app inventor hay que realizar los siguientes pasos:

- El diseño de la aplicación, en la que se seleccionan los componentes para su aplicación.
- El editor de bloques, donde se ira escogiendo los bloques que sean necesarios según la aplicación
- La aplicación aparecerá paso a paso en la pantalla del celular o tableta a medida que se añada piezas a la misma.

- Cuando se termina, se empaqueta la aplicación y se produce una aplicación independiente para instalar.
- Si no se tienes un celular o tableta Android, se puede construir las aplicaciones utilizando el emulador de Android , el software que se ejecuta en la computadora y se comporta como el celular.

El entorno de desarrollo de App aplicación es compatible con Mac OS X, GNU / Linux y sistemas operativos de Windows, y varios modelos de teléfonos Android populares. Las aplicaciones creadas con App Inventor se pueden instalar en cualquier teléfono Android.

En el diseñador escogeremos los componentes que vayamos a utilizar en nuestra aplicación, según nos interese.

1.9. COMPILADOR BASCOM AVR

Este compilador Basic en Windows que sirve para programar microcontroladores ATmega brinda herramientas, librerías, compilador, simulador y utiliza comando amigables para que el programador pueda realizar la respectiva programación.

Este compilador posee herramientas que reducen los tiempos en el desarrollo de la programación de forma drástica.

1.9.1. CARACTERÍSTICAS

- Programación estructurada con IF-THEN-ELSE-END IF, DO-LOOP, WHILE-WEND, SELECT- CASE.
- Código máquina compilado, mucho más rápido que los interpretados.
- Nombres de variables y etiquetas largos, hasta 32 caracteres de longitud.
- Variables Bit, Byte, Integer, Word, Long, Single y String.
- Los programas compilados trabajan con todos los microcontroladores AVR que tienen memoria RAM interna.
- Las instrucciones y comandos son en su mayoría compatibles con Microsoft VB/QB (Visual Basic/Q Basic).
- Mezcla Assembler y Basic en la misma fuente.

1.10. DISPOSITIVO BLUETOOTH HC06

El módulo a utilizar para la comunicación Bluetooth entre el sistema de control de la silla de ruedas y una Tablet, Celular o PC es el Módulo HC06 que a continuación se describirá.

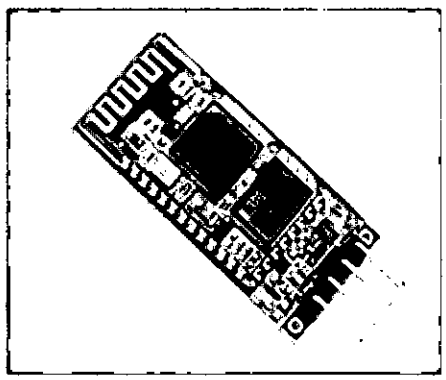


Fig. 1.26 Módulo Bluetooth HC06

Dispositivo	Alcance	Version	Ancho de banda	alimentacion	Baud
Módulo bluetooth HC-06	**10 mts**	**Bluetooth v2.0 + EDR.**	**1200bps a 1.3Mbps**	**3.3 / 5 v.**	**1200-1382400**
Módulo inalámbrico bluetooth TTL	**10 mts**	**Bluetooth v2.0**	**1.3Mbps**	**3.3v**	**1200-115200**

Tabla 1.6. Comparación de algunos dispositivos Bluetooth en el mercado

1.10.1 CARACTERÍSTICAS DEL MÓDULO HC06

- Módulo Bluetooth Slave HC-06
- Protocolo Bluetooth: Bluetooth especificación V2.0+EDR
- Frecuencia: 2.4Ghz ISM Band
- Rango de baudios ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Default: Slave, 9600 baud rate, N, 8,1. Pincode 1234
- Distancia Bluetooth: 10 metros
- Tamaño compacto

Una de las ventajas principales del módulo HC-06, además de su pequeño tamaño y sus buenas características de transmisión y recepción que le brindan un alcance muy amplio (por tratarse de un sistema local Bluetooth), es el bajo consumo de corriente que posee tanto en funcionamiento, como en modo de espera, es decir, alimentado con energía, pero sin conexión o enlace a otro dispositivo, por ejemplo, un móvil con SO Android.

Otra característica interesante de este módulo es que una vez que ha realizado un enlace con otro dispositivo es capaz de recordarlo en su memoria y no solicita validación alguna ("*1234*" por defecto), pero si se activa el pin 26 (*KEY*) hacia la tensión de alimentación, esta información se elimina y el módulo HC-06 solicitará nuevamente la validación del enlace. Otro detalle particular es que su tensión de alimentación de 3,3Volts y su bajo consumo (8mA en transmisión/recepción activa) lo transforman en un dispositivo ideal para trabajar con microcontroladores de la misma tensión de alimentación, logrando de este modo equipos portátiles que pueden ser alimentados durante muchas horas por baterías recargables o alcalinas AA, demostrando características excepcionales en aplicaciones médicas, o para actividades recreativas donde la fuente energética debe ser liviana y portátil.

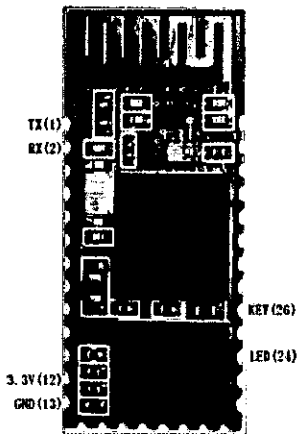


Fig. 1.27 Módulo Bluetooth HC06

1.10.2 CONECTANDO EL MÓDULO BLUETOOTH HC-06 CON MICROCONTROLADOR

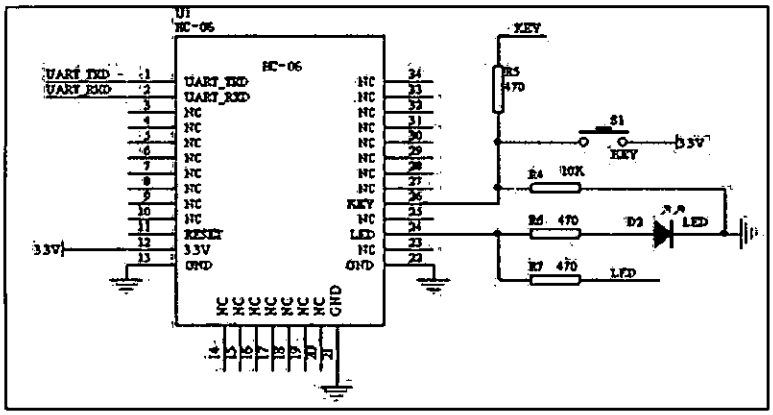


Fig. 1.28 Conexiones del Módulo Bluetooth HC06

El primer paso es reconocer que Módulo tenemos, para esto debemos conectar la alimentación del módulo a 3.3V, después debemos buscar el dispositivo Bluetooth ya sea con la PC o con un celular, el módulo HC-06 será encontrado con el nombre de "IINVOR".

Ahora debemos configurar nuestro Módulo, el HC-06 se puede configurar por medio de comandos AT y los valores que podemos modificar son el nombre del dispositivo, la contraseña (PIN) para realizar la conexión y el baudrate. Para que los comandos AT funcionen el Módulo no debe estar apareado con el dispositivo maestro, debe ser configurado por medio de un microcontrolador o mediante un convertidor usb-serial y la terminal serie en una PC.

Los comandos AT disponibles son los siguientes:

- **AT**: Sirve como test de comunicación, responde con **OK**
- **AT+VERSION**: Devuelve la versión del firmware del dispositivo, responde con OKlinvorV1.5
- **AT+NAME**: Cambia el nombre del dispositivo, por ejemplo AT+NAMEdispBT1 responde con OKsetname y ahora tendrá el nombre de dispBT1, el nombre es limitado a 20 caracteres.

- **AT+PINxxxx**: Cambia el pin de seguridad de 4 dígitos, podemos usar **AT+PIN0000** para setear el pin a 0000, responde con OKsetPIN, por default viene configurado 1234.
- **AT+BAUDx** : Modifica el baudrate del dispositivo, x puede tomar los siguientes valores

    1- 1200
    2- 2400
    3- 4800
    4- 9600 (Default)
    5- 19200
    6- 38400
    7- 57600
    8- 115200
    9- 230400
    A- 460800
    B- 921600
    C- 1382400

Se debe tomar en cuenta que el baudrate máximo que maneja una PC es de 115200, por lo que si estas configurando tu Módulo por medio de esta y escoges un baudrate mayor a 115200 perderás la comunicación completamente con el dispositivo, si esto llega a suceder solo podrás reconfigurarlo por medio de un microcontrolador capaz de manejar tal velocidad mayor a 115200. Si la velocidad no es primordial en tu diseño maneja la velocidad de 9600 por default o en caso necesario la de 115200 como máximo. Para 9600 baudios usamos AT+BAUD4 y responde OK9600

Una vez configurado el dispositivo lo podemos utilizar con un microcontrolador y realizar una comunicación serial de forma transparente.

Nota: Si se está trabajando solo con el chip, debe tomar en cuenta que el fabricante toma el pin TX(1) como la entrada de datos que serán enviados posteriormente por Bluetooth y RX(2) donde salen los datos recibidos por Bluetooth.

1.10.3 OBTENER MAC DEL MÓDULO BLUETOOTH HC06

Para realizar una aplicación en una Tablet o Celular en plataforma Android es necesario conocer la MAC del Módulo Bluetooth para lo cual explicaremos los pasos a seguir:

- En la Computadora Personal se debe tener Instalado Módulo Bluetooth USB y el Software Bluesoleit.
- Encender o Activar el Módulo Bluetooth
- En la PC realizar la búsqueda y conexión de clientes para realizar la vinculación luego de un tiempo aparecerá en la pantalla de la computadora el Número de MAC del Módulo HC06 parecido a: **00:11:10:20:02:06** el cual nos servirá para realizar la aplicación para tabletas o equipos celulares con Sistema Operativo Android. En la fig. se muestra una pantalla del Software Bluesoleil.

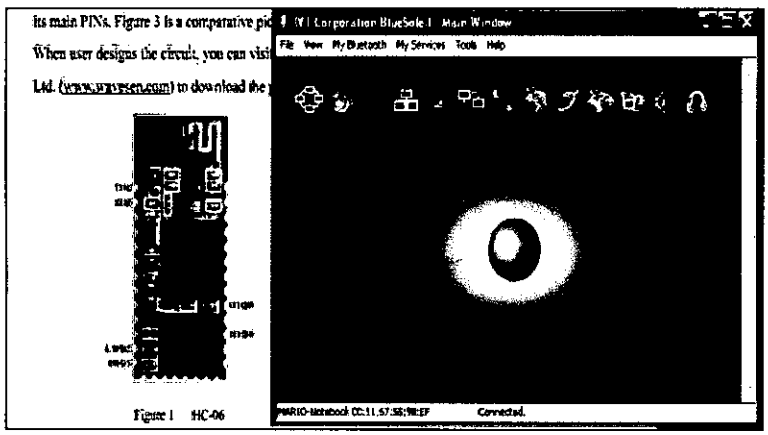


Fig 1.29 Conexión con software Bluesoleit

CAPITULO II

2. PROPUESTA DEL PROYECTO

TÍTULO: "Sistema de Control de silla de ruedas para personas parapléjicas y tetrapléjicas, usando Movimiento Traslacional, Voz, Bluetooth y S.O. Android".

2.1 JUSTIFICACION

Una silla de ruedas es una ayuda técnica que consiste en una silla adaptada con al menos tres ruedas, aunque lo normal es que disponga de cuatro.
Estas sillas están diseñadas para permitir el desplazamiento de aquellas personas con problemas de locomoción o movilidad reducida, debido a una lesión, enfermedad física (paraplejía, tetraplejía, etc) o psicológica.

La paraplejía es una enfermedad por la cual la parte inferior del cuerpo queda paralizado y carece de funcionalidad. Normalmente es resultado de una lesión medular o de una enfermedad congénita como la espina bífida. Una polineuropatía puede tener también como consecuencia la paraplejía. Si los brazos se ven afectados también por la parálisis la enfermedad se denomina tetraplejía.

En el mercado existen Sillas de ruedas mecánicas que son manejadas por lo usuarios con las manos para mover las ruedas, las sillas de ruedas motorizadas son de costo elevado y no están al alcance de todos las personas que lo necesitan.

Los usuarios de estas sillas motorizadas actualmente se manejan mediante un sistema mecánico para lo cual se utiliza unos mandos tipo Joystick.

Si los usuarios son ancianos o personas con discapacidad en las manos no podrían controlar ese tipo de sillas de ruedas motorizadas, por lo cual se plantea un sistema que permita controlar por voz a la silla y además de un sistema alternativo que permita controlar la silla con sensores tipo acelerómetros que se podrían colocar en las muñecas de las manos, cuello, tobillos para así controlar la silla.

Por lo tanto se implementará Sistema de Control de motores utilizando el sistema operativo para dispositivos móviles más con mayor presencia en el mercado, ANDROID y será probada en una silla de ruedas normal para darle prestaciones diferentes inclusive que las motorizadas y a un costo más asequible y sobre todo como parte de la ayuda que deseamos brindar a las personas con paraplejia y tetraplejia.

2.2. OBJETIVOS

Diseñar e Implementar un Sistema de control de una Silla ruedas que permita ser controlada por una persona parapléjica con el mínimo esfuerzo usando el movimiento traslacional y/o voz.

Implementar el sistema de comunicación entre la silla de ruedas y el usuario usando la Tecnología Bluetooth y Teléfonos Celulares con Sistema Operativo Android.

2.3. METODOLOGÍA

- Primero: Diseño de sistema de Comunicación Bluetoooh.
- Segundo: Diseño e implementación de tarjetas de adquisición de señales de los acelerómetros
- Tercero: Construcción de la Silla de ruedas.
- Cuarto: Instalación de Motores Eléctricos y Sistema de Baterías
- Quinto: Diseño de tarjetas Electrónicas Controladoras de los Motores y su simulación.
- Sexto: Implementación de Tarjetas Electrónicas y Pruebas con carga.
- Séptimo: Diseño de sistema de Comunicación Bluetoooh.
- Octava: Instalación de sensores acelerómetros para movimiento traslacional, Pruebas de control de movimiento de la silla haciendo uso de los sensores acelerómetros.
- Noveno: Pruebas de control de la silla mediante la voz.
- Decimo: Pruebas finales de todos los sistemas en conjunto.
- Onceavo: Elaboración Informe Final.

CAPITULO III

3. DISEÑO DEL HARDWARE PARA CONTROL DE SILLAS DE RUEDAS

3.1. CARACTERÍSTICAS ADICIONALES A UNA SILLA DE RUEDAS COMÚN

La silla de ruedas que se propone debe tener las siguientes características ver fig. Siguiente:

- Configurable con celular la manera de controlar la silla
- Sistema de Comunicación Bluetooth
- Control de movimiento usando Celular con Sistema Operativo Android (teclado, acelerómetro y voz)
- Control de movimiento traslacional usando acelerómetros como sensores de movimiento
- Control de movimiento usando la Voz
- Control de movimiento manual usando Joystick
- Sistema de localización con GPS.

Fig. 1.30 Características adicionales para una silla de ruedas común.

3.2 DISEÑO DE BLOQUES DEL SISTEMA ELECTRÓNICO

El diseño de bloques se muestra en la figura, el componente principal es el microcontrolador Atmega32, el cual tienes las características suficientes para realizar el sistema de control de la silla de ruedas, las cuales se detallan en el Capítulo II.

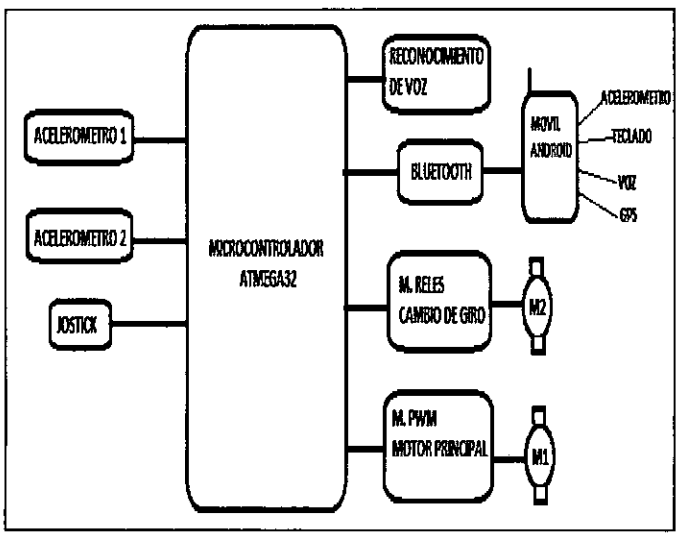


Fig. 1.31 Diseño de bloques de Control de Silla de ruedas

3.3 DESCRIPCIÓN DEL HARDWARE DEL SISTEMA

3.3.1 SENSOR DE MOVIMIENTO ACELERÓMETRO

Esta tarjeta posee un acelerómetro de 3 ejes ADXL335 de Analog Devices, ver Fig.3.3 Posee lo último en tecnología en la línea de este tipo de sensores analógicos. El ADXL335 es poco ruido y bajo consumo de potencia, solamente 320uA. El sensor tiene un rango de sensibilidad de +/-3g.

No posee regulación de voltaje integrada por lo que debe alimentarse con voltajes en el rango de 1.8 a 3.6VDC.

La tarjeta viene totalmente ensamblada, probada y con todos los componentes externos instalados. Incluye capacitores de 0.1uF para fijar el ancho de banda de cada eje en 50Hz. Ver Fig. siguiente.

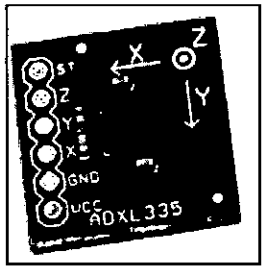


Fig. 1.32 Acelerómetro

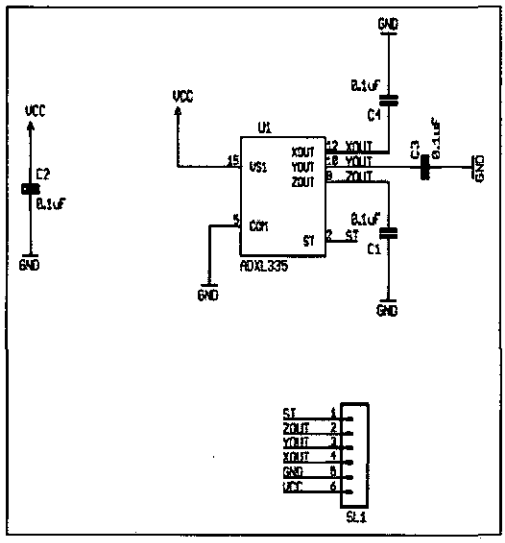


Fig. 1.33 Esquema eléctrico del Acelerómetro

Cuando un condensador de 0,1 μF, es colocado cerca de las clavijas de alimentación el ADXL335 se desacopla adecuadamente el acelerómetro del ruido de la fuente de alimentación. Sin embargo, en aplicaciones en las que el ruido está presente en la frecuencia de 50 kHz de reloj interno, se requiere un cuidado adicional en derivación fuente de alimentación porque este ruido puede causar errores en la medición de la aceleración.

Si se necesita desacoplamiento adicional, un 100 Ω (o menor) resistencia o núcleo de ferrita se pueden insertar en la línea de suministro. Adicionalmente, un condensador de derivación de bulto de mayor tamaño (1 mF o mayor) puede añadirse en paralelo a los DC . Asegúrese de que la conexión de la tierra ADXL335 al suelo fuente de alimentación es de baja impedancia porque el ruido transmitido a través del suelo tiene un efecto similar al ruido transmitido a través VS.

El ajuste del ancho de banda de usar cx, cy, cz y el adxl335 tiene disposiciones para la banda que limitan la XOUT , YOUT y pins ZOUT . Los condensadores se deben agregar a estos pines filtro de paso bajo para el antialiasing y reducción de ruido para implementar.

Una capacidad mínima de 0,0047 uF para CX , CY y CZ se recomienda en todos los casos .

La conexión es como la que se muestra en la Fig.3.5.

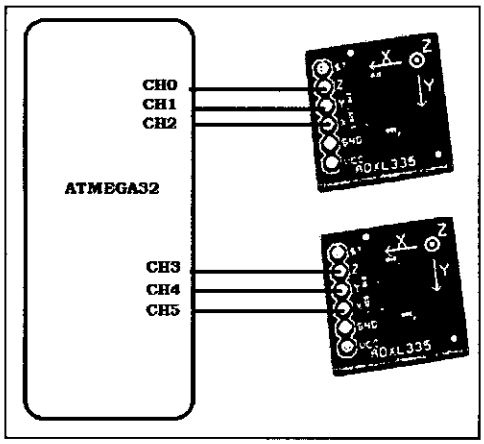


Fig. 1.34 Conexión de los acelerómetros con el microcontrolador

3.3.2. MÓDULO DE COMUNICACIÓN BLUETOOTH

El módulo Bluetooth HC-06 viene configurado de fábrica para trabajar como esclavo, es decir, preparado para escuchar peticiones de conexión. Se alimenta con 5V, la velocidad de comunicación con el microcontrolador es de 9600 Baud. El circuito se muestra en la Fig.3.6.

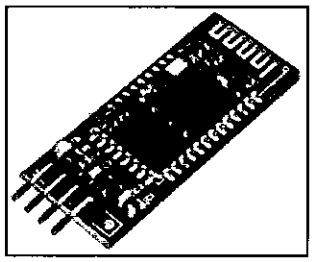


Fig. 1.35 Módulo Bluetooth HC06

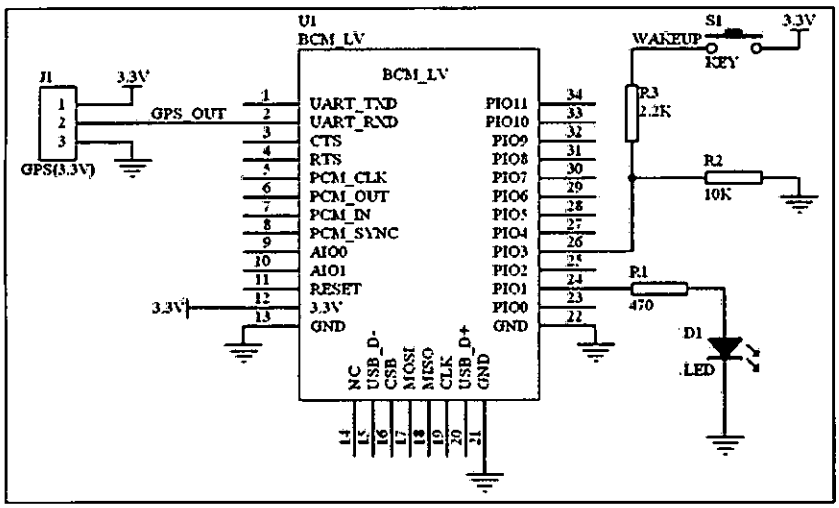


Fig.1.36 Circuito del Módulo HC06

Módulo en sus pines TX y RX envía y recibe datos mediante un protocolo serial pero con un nivel lógico TTL (igual que un microcontrolador). La computadora en su puerto serial mediante la norma RS232 maneja niveles de tensión de +/-10V por lo que si conectamos directamente el módulo al puerto serial quedará inservible. Tenemos que poner el medio un transceptor como el MAX232 el cual convierte el TTL en RS232 y el RS232 en TTL de esta forma podemos comunicar nuestro módulo.

A continuación de indica la conexión del Módulo Bluetooth con el microcontrolador ATMEGA32. .Fig.3.8

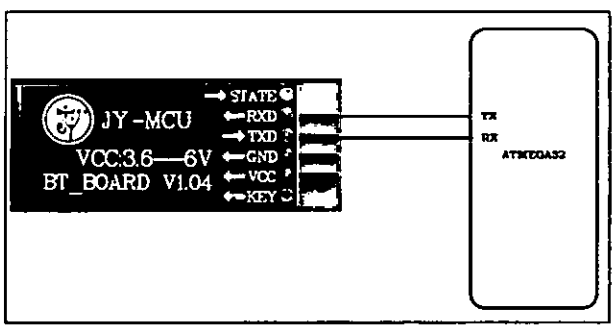


Fig.1.37 Conexión del Módulo Bluetooth HC06 con el microcontrolador

3.3.3 MÓDULO DE RECONOCIMIENTO DE VOZ

Dispositivo	interfaces	Ancho de banda	alimentacion
Voice Recognition Module V2	3.5mm mono-channel microphone connector + microphone pin interface	15 activaciones	4.5-5.5V
EasyVR Module	UART	32 activaciones	3.3V – 5V

TABLA 1.7 Algunos Módulo de reconocimiento de Voz en el Mercado

El módulo de reconocimiento de voz de Elechouse es capaz de grabar y reconocer 15 comandos de voz grabados por uno mismo, y grabados en 3 grupos de 5. Con este módulo conectado a un microcontrolador se puede empezar a controlar la silla de ruedas.

Este módulo está diseñado para reconocer el volumen, la modulación y la entonación de las palabras para saber exactamente que comandos desea ejecutar.

Los comandos que tienen grabados son:

- Avanza
- Retro
- Izquierda
- Derecha
- Parar

Estos comandos están grabados en el primer grupo. Cada comando maneja una salida digital, por lo tanto se tiene 5 salidas digitales que se activan según comando.

Estas 5 salidas digitales están conectados hacia el microcontrolador para el cual vienen hacer entradas digitales, después de reconocido el comando el microcontrolador envía comando de reset para el Módulo de reconocimiento de Voz. En la Fig.3.9

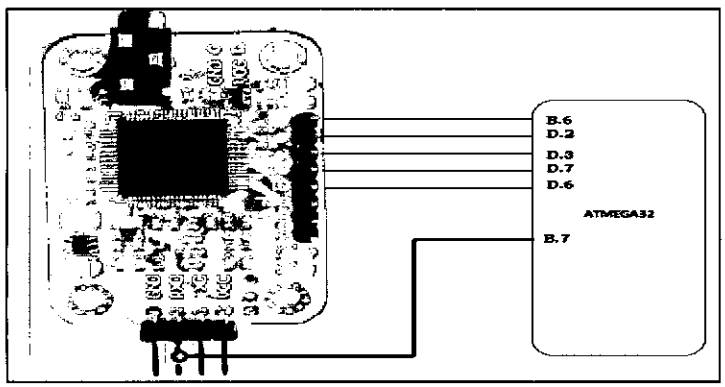


Fig.1.38 Conexión del Módulo de reconocimiento de voz al microcontrolador

El Módulo de Reconocimiento de voz se programa a través de un adaptador USB a Serial y los comandos se graban en tres grupos de 5 comandos cada uno (Fig.3.10). Para más detalle ver apéndice.

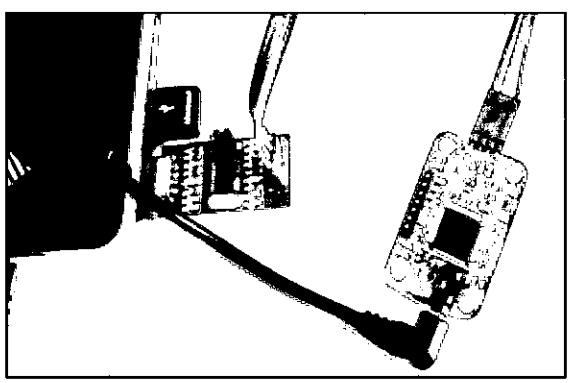


Fig.1.39 Conexión de Módulo de Reconocimiento de Voz por puerto USB

ESPECIFICACIONES:

Voltaje de Alimentación: 4.5 ~5.5VDC

Corriente: <40mA

Interfaz digital: 5V TTL.

Interfaz análoga: conector 3.5mm para micrófono.

Dimensiones: 30mm x 47.5mm

Precisión de reconocimiento: 99% (bajo entorno ruidos reducidos).

3.3.4. MÓDULO JOYSTICK

Este módulo permite agregar funcionalidad para regular movimientos en 2 ejes. Se conecta el módulo a dos entradas análogas, del microcontrolador y se tiene un control preciso de la posición.

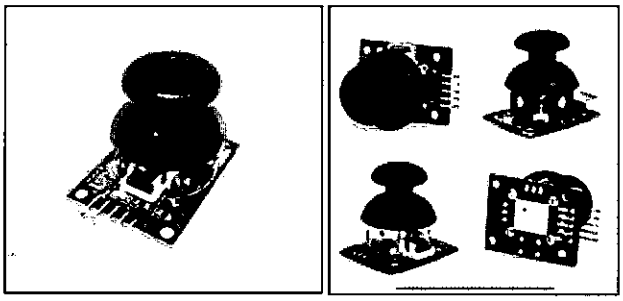


Fig.1.40 Módulo Joystick

La conexión con el microcontrolador es como la que se muestra en la Fig.3.12

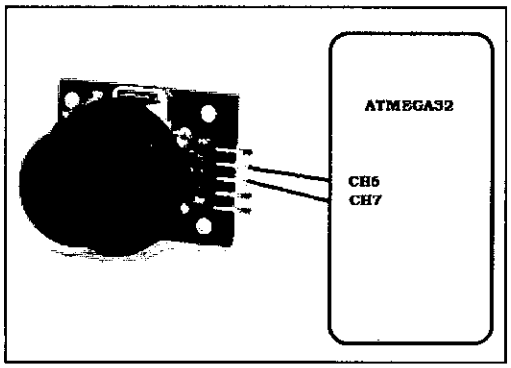


Fig.1.41 Conexión del Joystick con el microcontrolador

3.3.5. MÓDULO DE POTENCIA PWM:

El módulo de potencia está basado en Mosfets, Este módulo consta de 4 chips BTS7960 de alta performance y alta capacidad de corriente se utiliza para controlar el Motor que mueve la silla para adelante o de retroceso.

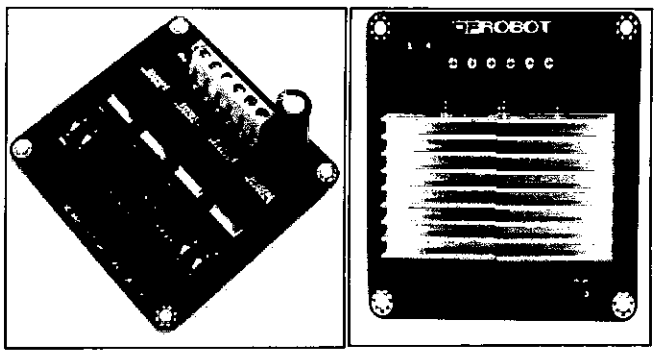


Fig.1.42 Módulo de Potencia BTS7960

Con este módulo se puede controlar hasta dos motores en simultáneo usando solamente 4 pines digitales. Este módulo tiene dos puentes H. El módulo es capaz de entregar hasta 15/13.8V por canal, con una excelente respuesta y tiempos de frenado. Incorpora leds indicadores, disipadores de aluminio en las partes posteriores y borrneras para asegurar un buen conexionado.

En la Fig.3.14 se puede observar la conexión del microcontrolador con el módulo de potencia BTS7960 y la Batería de 12V.

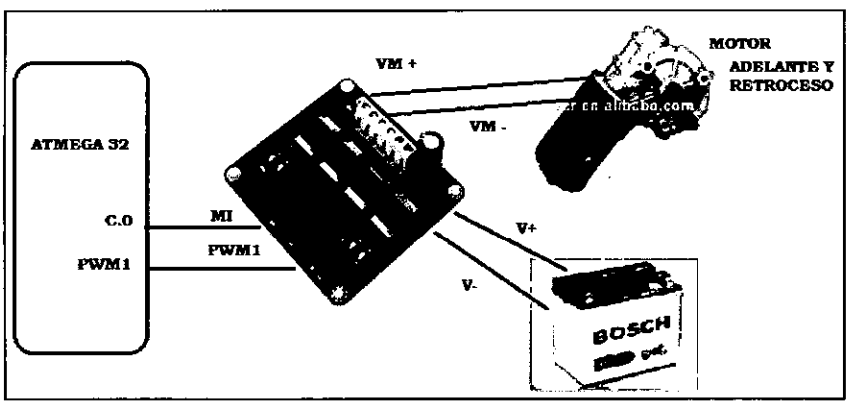


Fig.1.43 Conexión del módulo de potencia BTS7960 con Motor y Batería 12V

3.3.6. MÓDULO DE CAMBIO DE GIRO DE LA SILLA

Para lograr que la silla de ruedas realiza giros hacia la izquierda o derecha se utilizo un registro basado en relés que soportan una carga de 10 A. El circuito es el que se muestra en la Fig.3.16 Donde las señales de control se conecta a los puertos C.6 y C.7 del microntrolador ATMEGA32.

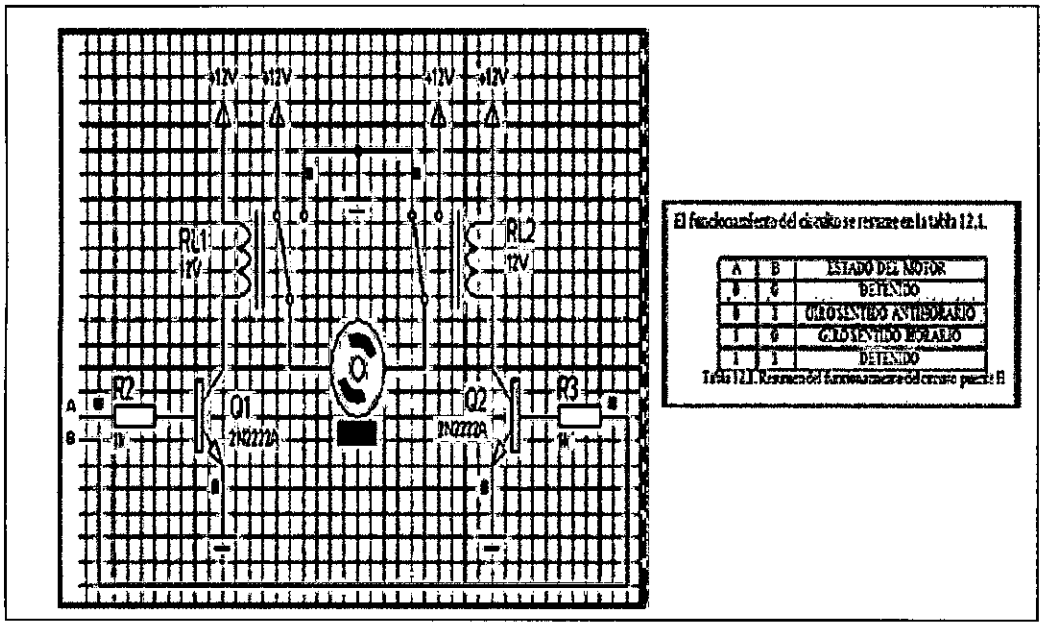


Fig.1.44 Circuito de Cambio de giro para la silla de ruedas.

Cuando llega el nivel lógico 1 al terminal A la corriente fluye a través de la resistencia R2 y llega la base polarizando al transistor 2N222, dejando este transistor pasar la corriente, energizando la bobina del relé, haciendo que su contacto que esta normalmente cerrado habrá y cierre el circuito que hace girar el motor en un sentido. Caso similar sucede cuando el terminal B esta en nivel lógico 1.

CAPITULO IV

4. DISEÑO DEL SOFTWARE PARA CONTROL DE SILLAS DE RUEDAS

4.1. DESARROLLO DEL SOFTWARE PARA DISPOSITIVOS MÓVILES

Para el desarrollo del Software para el dispositivo movil con sistema operativo Android se ha utilizado el appinventor como se muestra en la Fig, se muestra la pantalla principal de la aplicación indicando las funciones de cada uno de los botones, señalizador y cajas de texto de la aplicación.

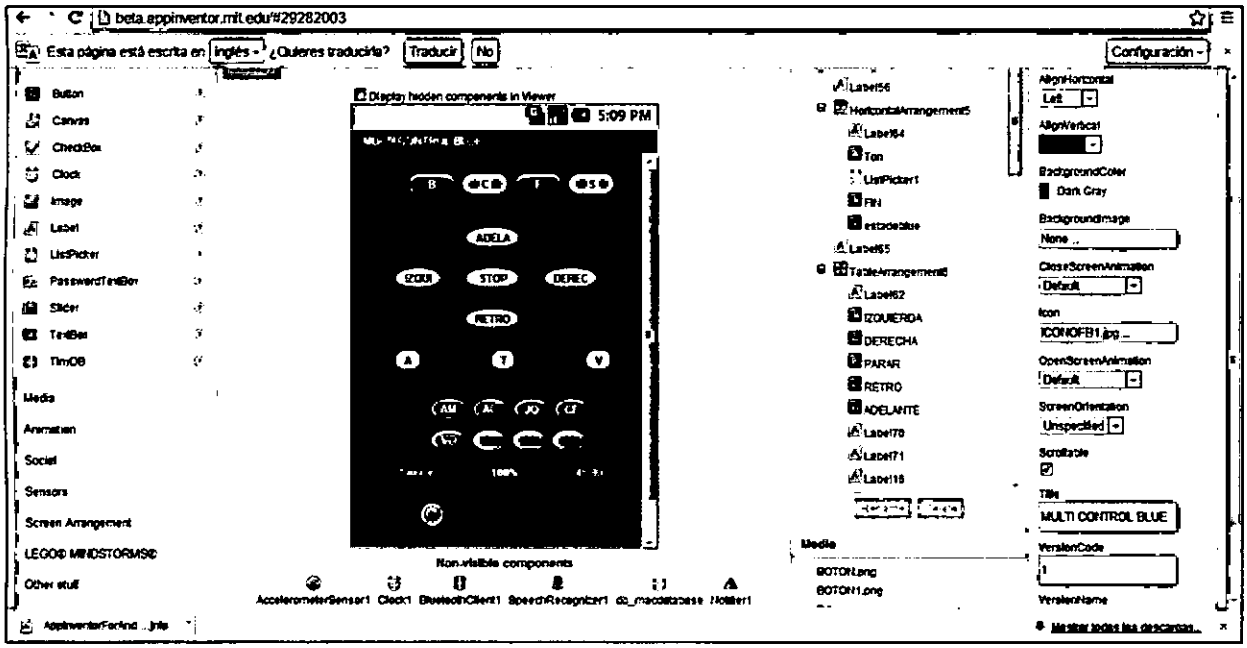


Fig.1.45 Pantalla de Diseño de la aplicación

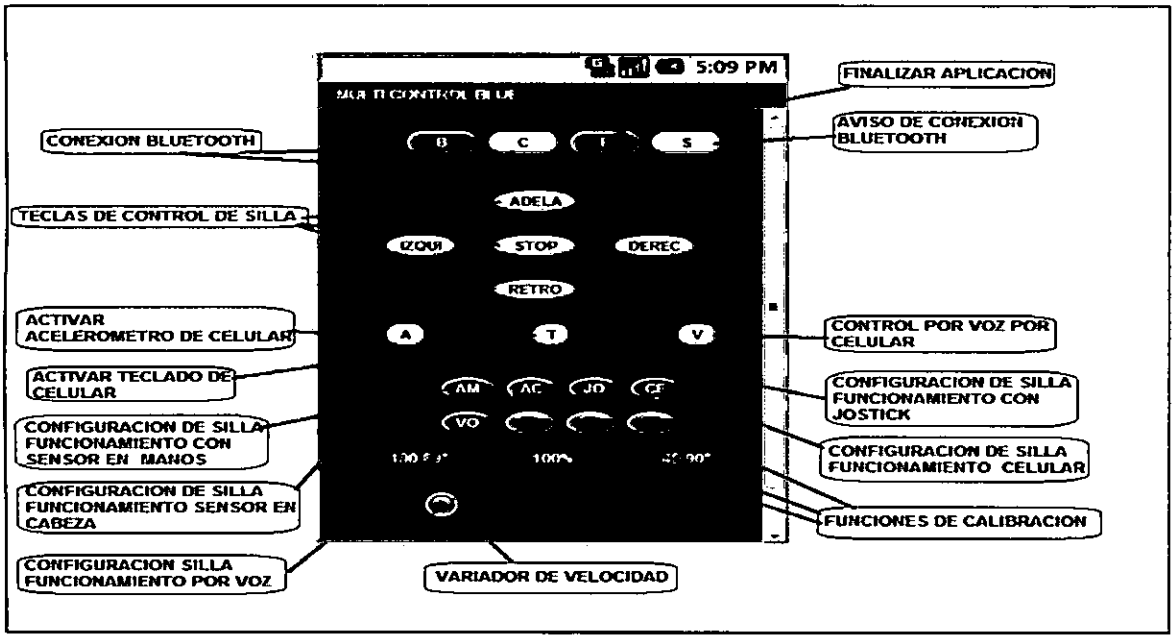


Fig.1.46 Descripción funcional de la pantalla principal de la aplicación

4.2. DESCRIPCIÓN DE LA CODIFICACIÓN GRAFICA DE LA APLICACIÓN

A continuación en la Fig. se muestra las diferentes partes y funciones que componen la aplicación en el dispositivo móvil.

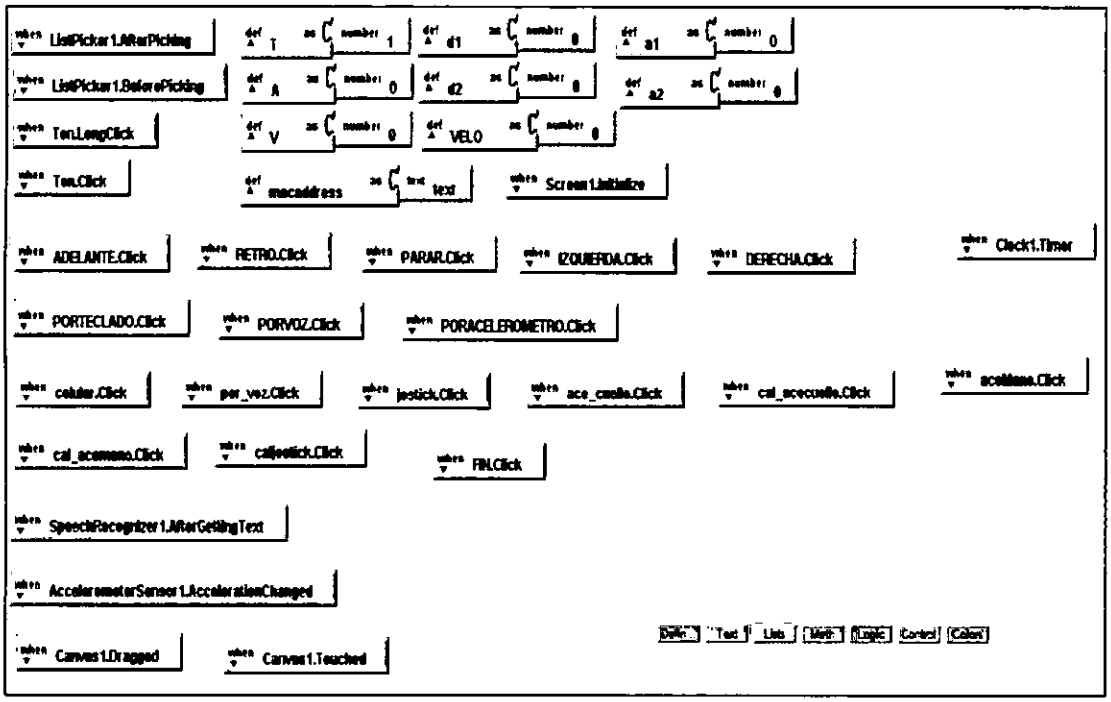


Fig. 1.47 Componentes de la Aplicación para dispositivos móviles

Ahora se empezara describiendo los componentes más importantes de la aplicación, para más detalle se tiene el código fuente en el CD de la tesis.

4.3. CARGA INICIAL:

La aplicación al iniciar la ejecución cargara los nombres o macaddress de los diferentes módulos Bluetooth con lo que el equipo se ha conectado y presentará un listado de los mismos, la figura siguiente muestra el código del programa.

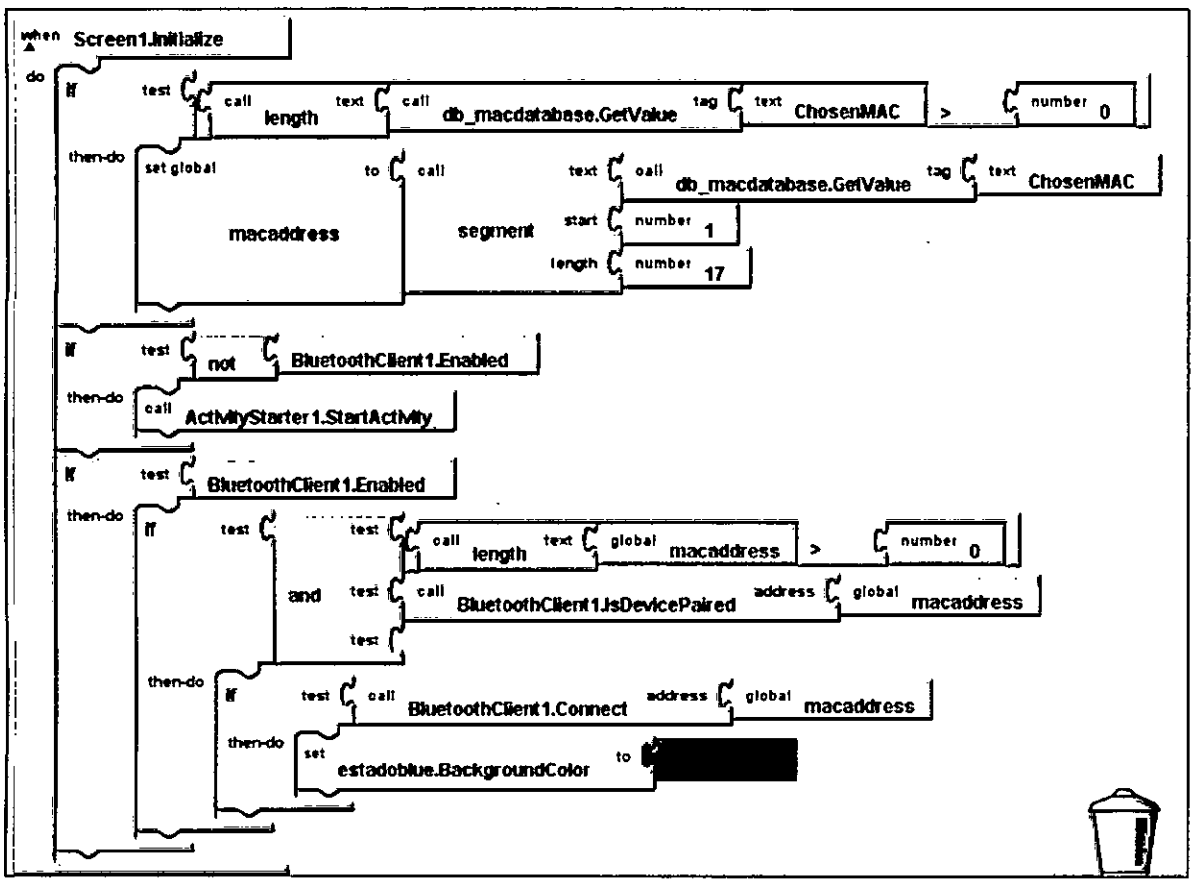


Fig.1.48 Código de carga Inicial

4.4. BOTONES DE DIRECCIÓN

La aplicación tiene 5 botones o teclas para cambiar la dirección de la silla los cuales son: adelante, retroceso, izquierda, derecha y parar.

Fig.1.49. Botones de dirección

A continuación se describirá el código de los botones, para todos los botones se tiene la misma lógica solo se cambia el comando a enviar. Ver Fig.

Primero se verifica que este en modo Teclado para lo cual se verifica que la variable T se igual a 1, luego se verifica que se esté conectado vía Bluetooth, y luego se envía comando según tecla presionada para el caso "ADELANTE" se envía hacia la tarjeta controladora el texto "* M100", y el botón se pone color verde y los otros se ponen en color gris. Para los demás botones solo se cambia el comando. Se muestra los comandos para las diferentes direcciones.

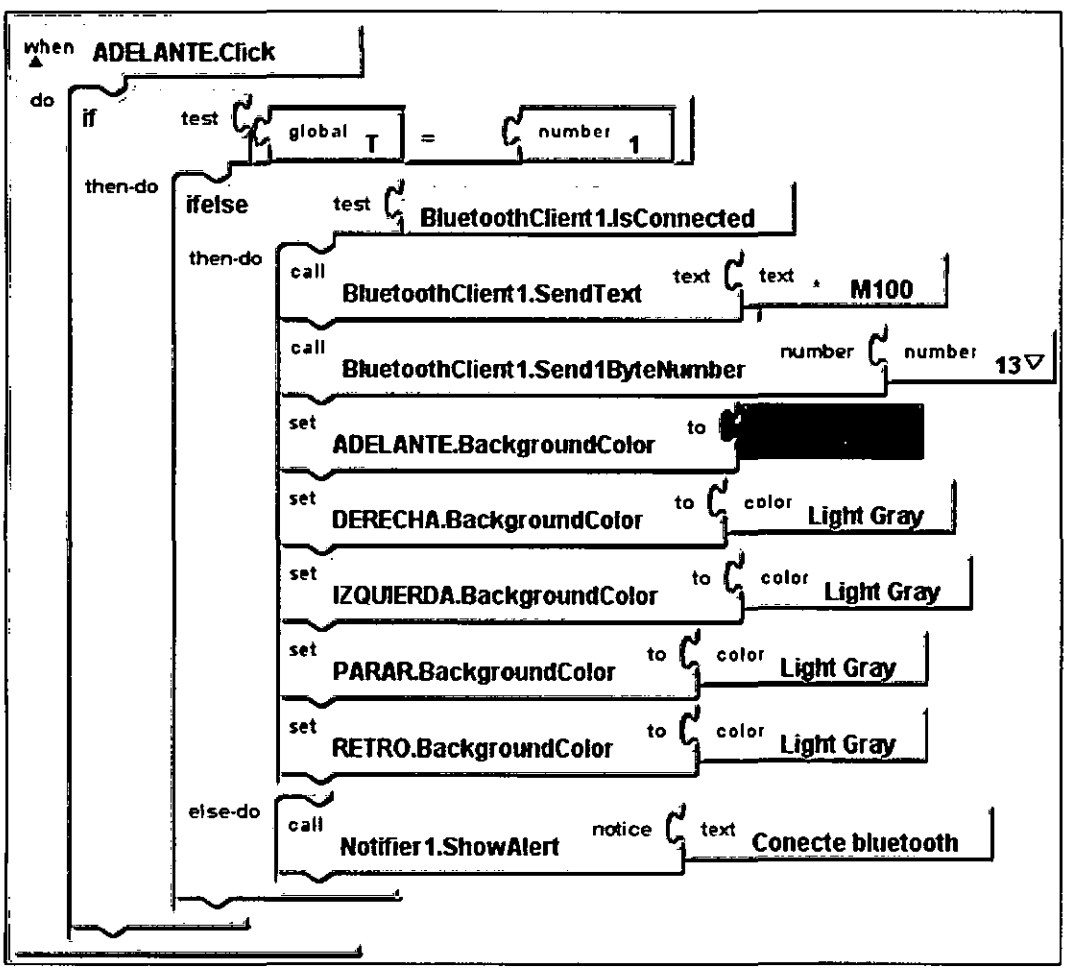


Fig. 1.50 Código para Botones de Dirección

4.5 BOTONES DE CONFIGURACIÓN USANDO CELULAR:

Usando el dispositivo móvil se puede configurar para que utilice los recursos del mismo equipo y la silla pueda ser controlada por teclado, por voz o usando su acelerómetro.

Fig. 1.51 Botones de configuración usando recursos del dispositivo móvil

La lógica para los tres botones es la misma, como se verifica en la figura siguiente, al presionar uno de los botone T, V o A, se asigna un valor de 1 una variable y las otras a cero y se le asigna a su vez el color amarillo.

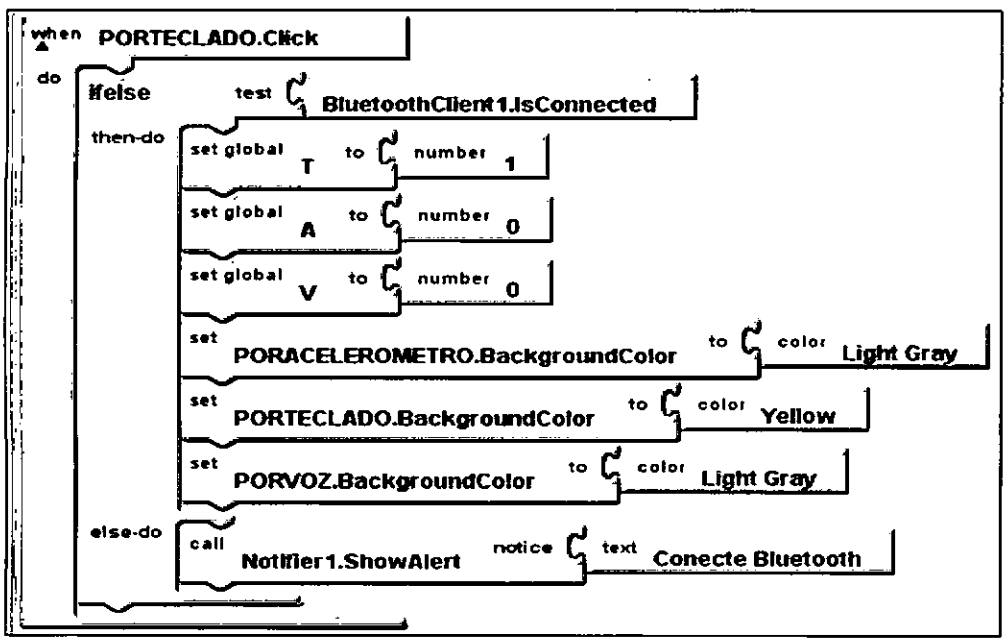


Fig.1.52 Botones de funcionamiento con celular por teclado, acelerómetro o por voz

4.6. BOTONES DE CONFIGURACIÓN DE FUNCIONAMIENTO DE LA SILLA CON SENSORES EXTERNOS:

Mediante el dispositivo móvil podemos configurar para que la silla funcione con sensores externos al celular, de esta manera la silla se podrá controlar mediante reconocimiento de Voz, Acelerómetros o Joystick, además tienes botones de ayuda para realizar la calibración de sensores para el diseñador.

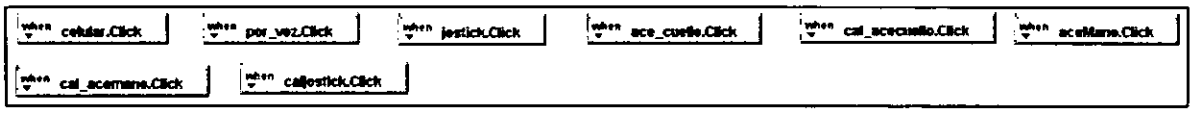


Fig.1.53 Botones de configuración de la silla usando sensores externos

La lógica de todos los botones es la misma y es como se muestra en al Fig., donde solo se en envía un comando por Bluetooth hacia la tarjeta controladora el texto * C100 que es para configurar funcionamiento por celular., en la Fig. se muestra el total de comandos a utilizar.

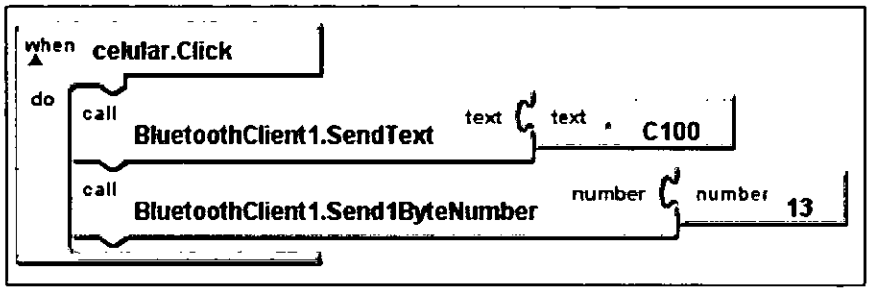


Fig. 1.54 Botón de Configuración de funcionamiento de la silla con sensores externos.

4.7. CALCULO DE ANGULO DE MOVIMIENTO CON ACELERÓMETRO DE EQUIPO MÓVIL

El equipo móvil posee un acelerómetro interno por tal motivo, se puede utilizar este sensor para calcular el ángulo de rotación del equipo y designar que ángulos corresponden para el movimiento izquierda, derecha, adelante, retroceso y parar. Ver Fig.

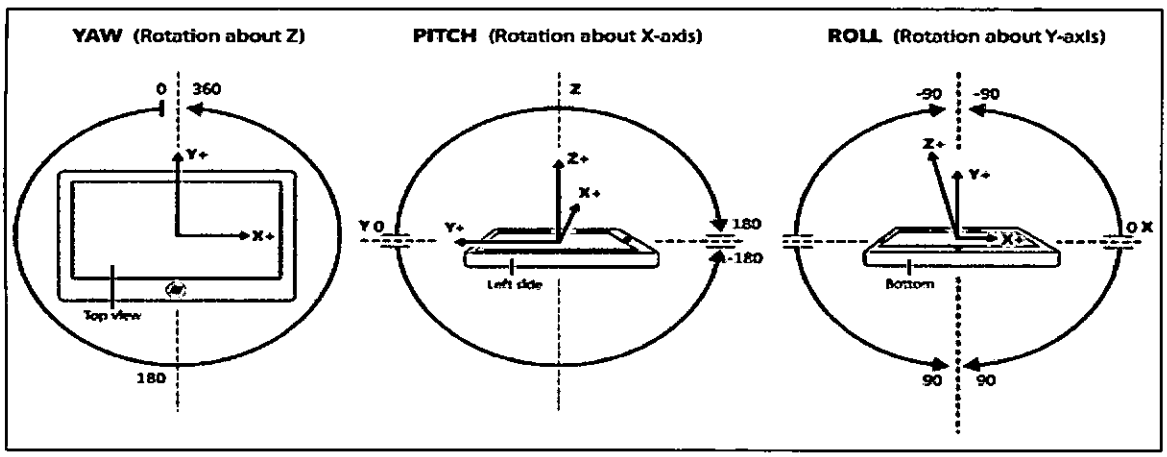


Fig.1.55 Ángulos de Rotación de un equipo Móvil con acelerómetro

Para calcular los ángulos, ver Fig. 4.11

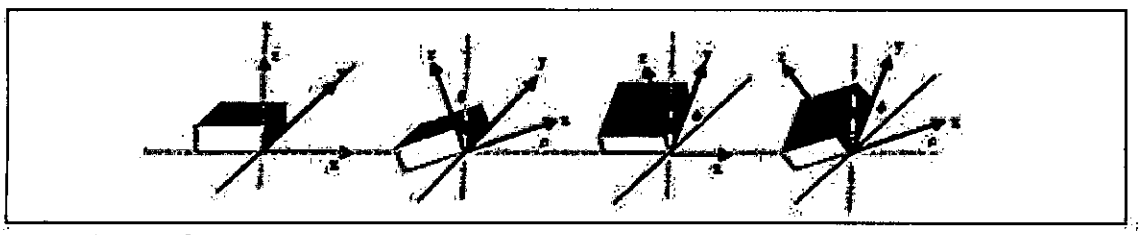


Fig. 1.56 Ángulos de un equipo Móvil con acelerómetro.

En este caso, se cuenta con tres rotaciones que se llamarán pitch (ρ), roll (φ) y theta (θ), donde pitch es definido como el ángulo formado entre "**x**" y la referencia; roll, el ángulo formado en "**y**" con respecto a la referencia, y theta es el ángulo con respecto a la gravedad, de esta forma es posible ser más precisos en la obtención del ángulo. Haciendo uso de las ecuaciones mencionadas anteriormente se concluyen las ecuaciones de la Fig., para definir el ángulo de rotación de cada eje.

$$\rho = Arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

$$\varphi = Arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

$$\theta = Arctan\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right)$$

Manteniendo "**z**" como nuestra referencia, solo nos interesan los ángulos pitch (ρ), roll (φ), para obtener los ángulos de rotación se sigue el código que se muestra en la fig. siguiente.

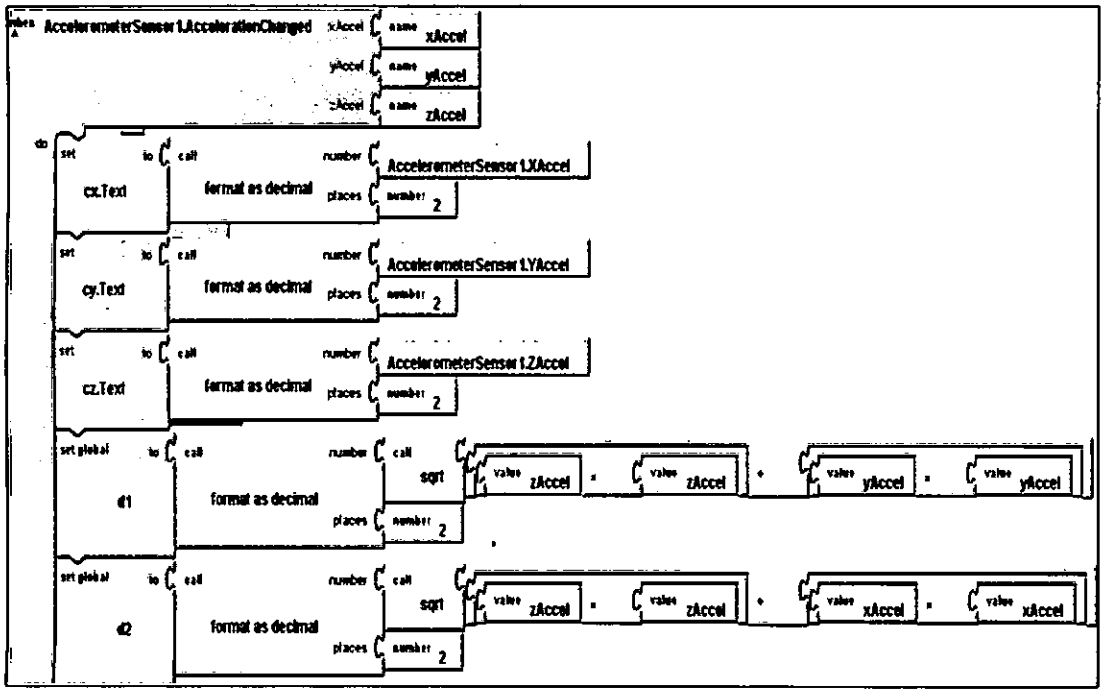


Fig.1.57 Se lee el valor de los 3 ejes, se determina el denominador, para cálculo de ángulo

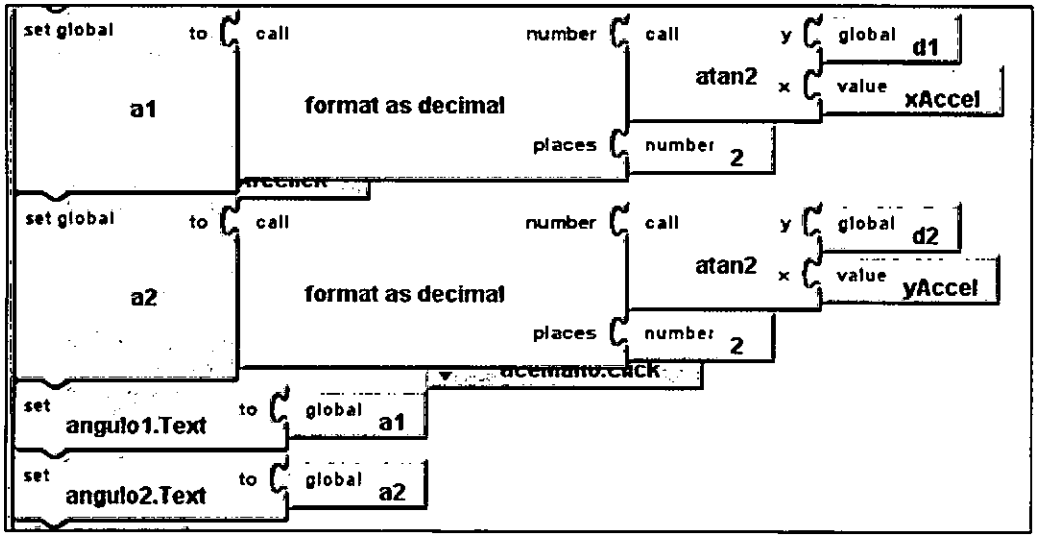


Fig. 1.58 Se determina el valor del ángulo

Una vez determinados los ángulos donde: "a1" es (ρ), "a2" es (φ), se procede a determinar los rangos de estos para los diferentes movimientos que va realizar la silla. Este procedimiento se muestra en las figuras.

Movimiento a la Izquierda cuando (ρ) < 85, Ver Figura., se envía comando " *M400"

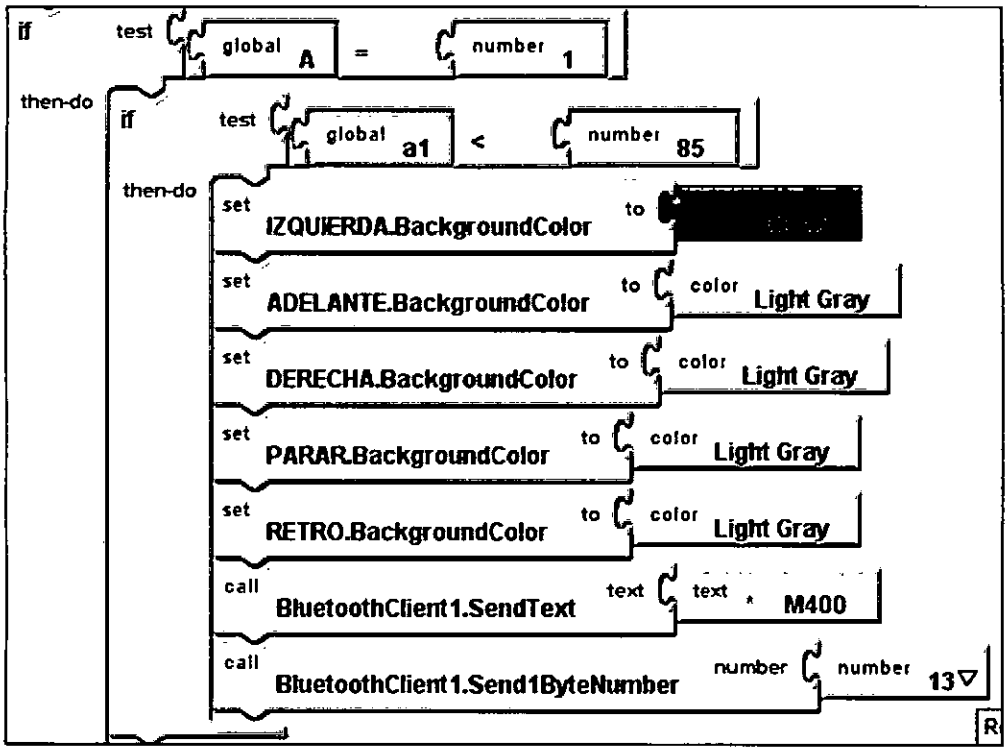


Fig.1.59 Código para movimiento a la Izquierda

Movimiento a la derecha cuando (ρ) >100, Ver Figura., se envía comando " *M200"

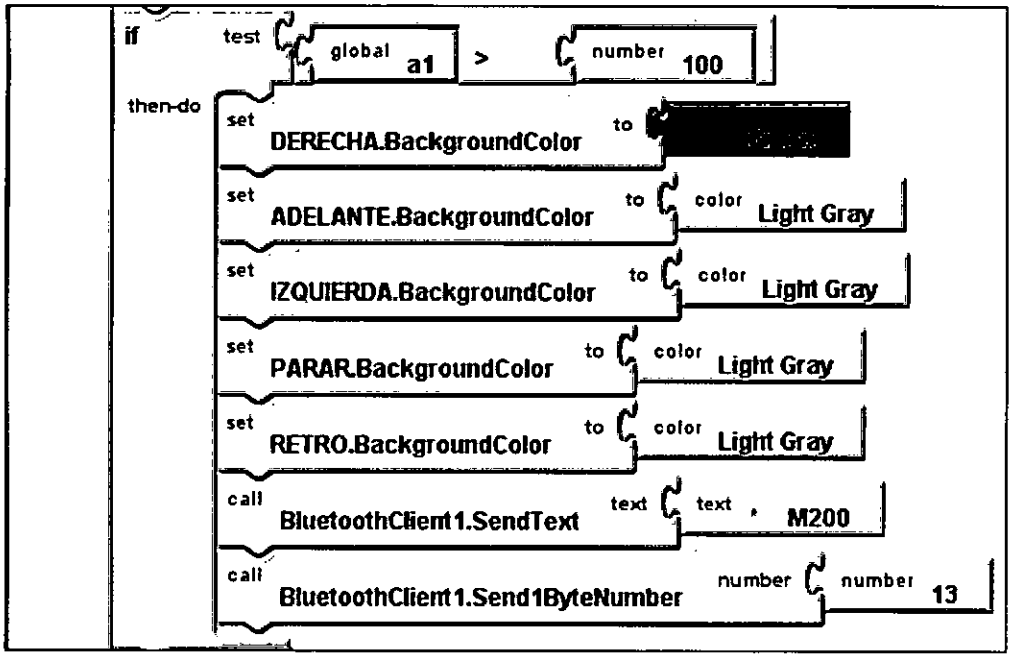


Fig.1.60 Código para movimiento a la Derecha

Movimiento hacia adelante cuando (φ)>95, Ver Figura., se envía comando " *M100"

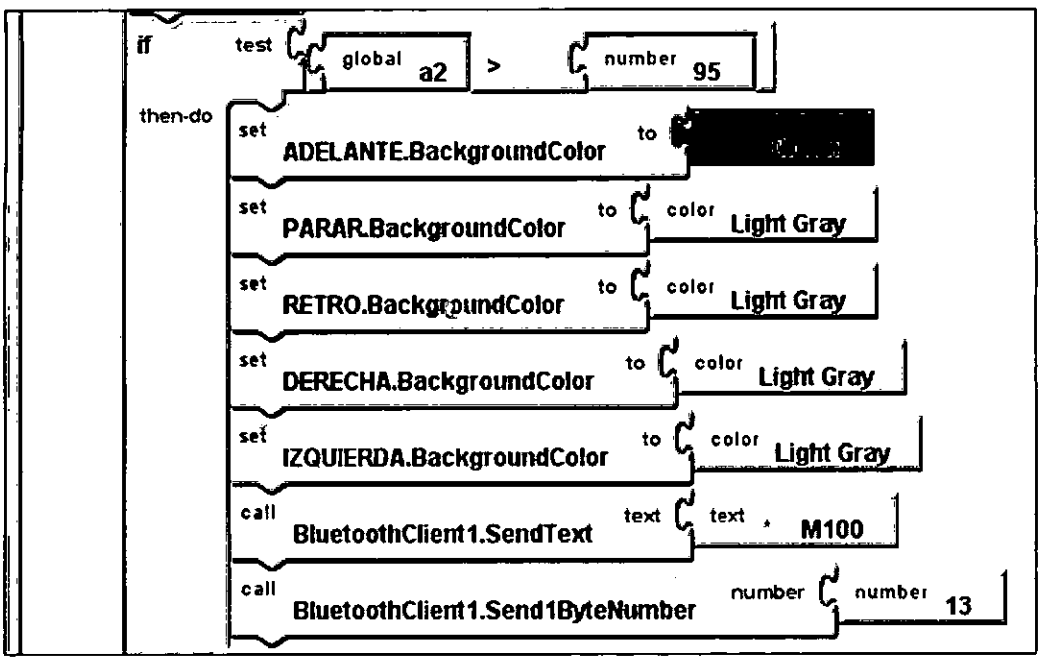


Fig. 1.61 Código para movimiento adelante

Movimiento de retroceso cuando (φ)<70, se envía comando " *M300"

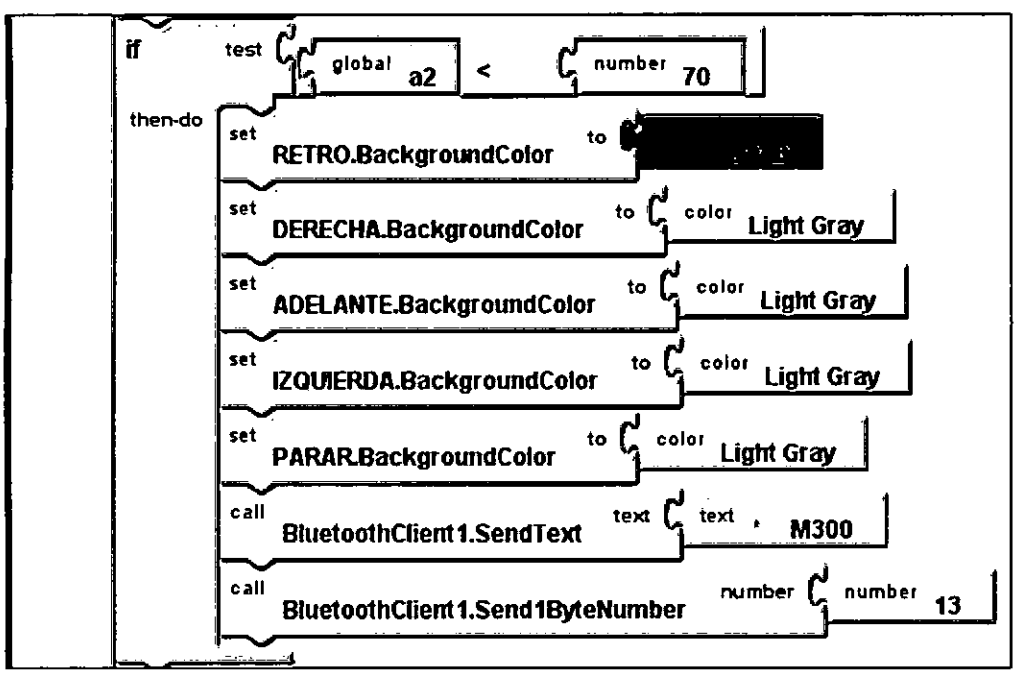


Fig.1.62 Código para movimiento de retroceso

Parar cuando 70<(φ)<95, 85<(ρ)<100, se envía comando " *M500"

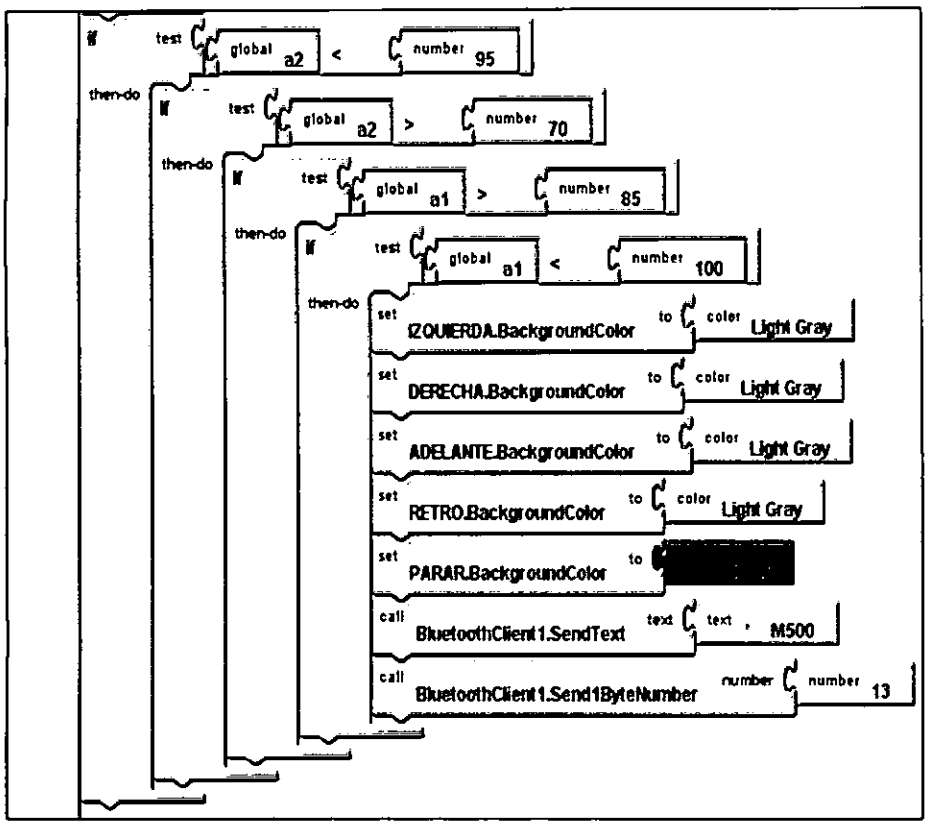


Fig.1.63 Código para movimiento de parada

4.8. CONTROL DE MOVIMIENTO CON RECONOCIMIENTO DE VOZ DEL EQUIPO MÓVIL

Para el reconocimiento de voz se utiliza la aplicación que viene por defecto en el equipo (Fig.), y se enlaza con nuestra aplicación como se muestra en la Fig.

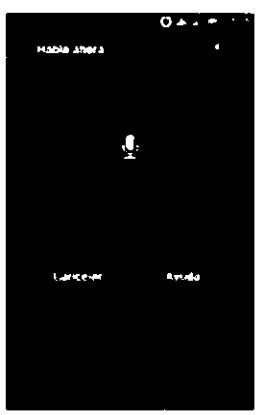


Fig. 1.64 Aplicación por defecto para reconocimiento de voz.

Para el reconocimiento de voz en el appinventor se utiliza el módulo de SpeechRecognizer, el cual una vez captada la voz la compara con una palabra según el movimiento a realizar "ADELANTE, ATRÁS, IZQUIERDA, DERECHA, PARAR" y si coinciden se envía el comando respectivo según figura.

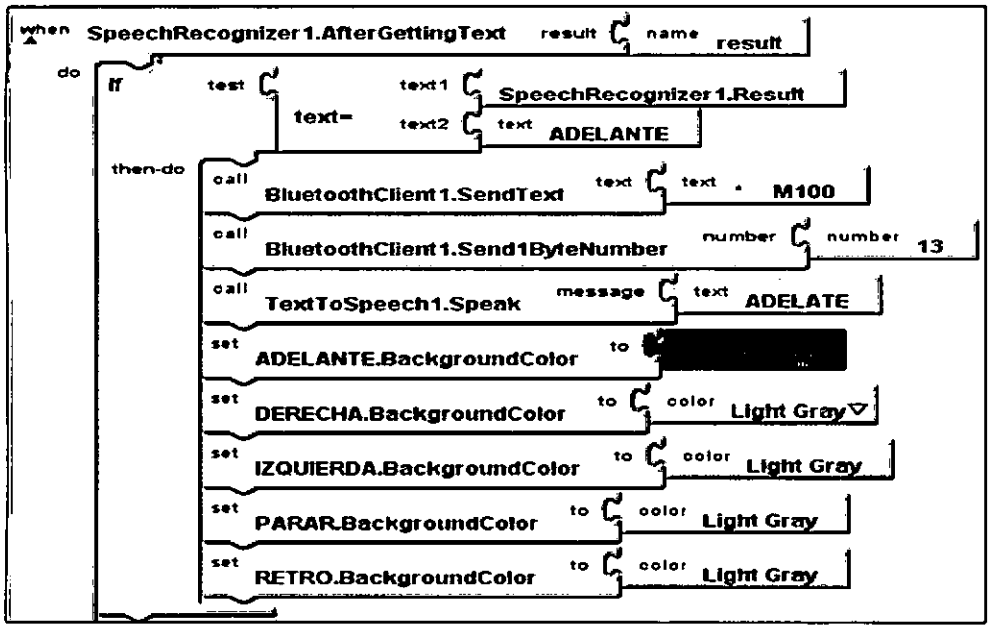


Fig. 1.65 Reconocimiento de Voz, Movimiento "ADELANTE"

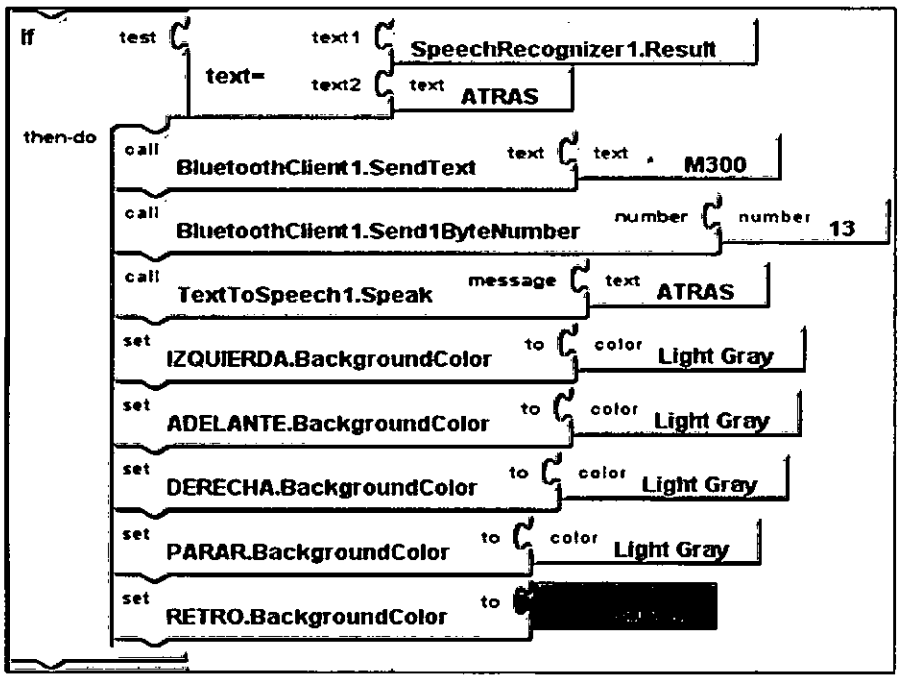


Fig.1.66 Reconocimiento de Voz, Movimiento "ATRAS"

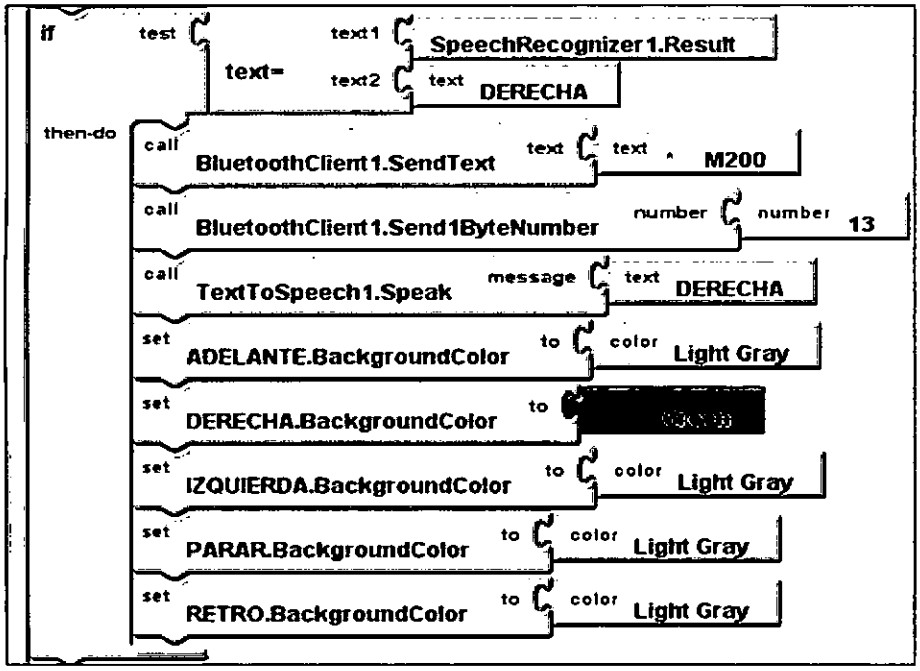


Fig. 1.67 Reconocimiento de Voz, Movimiento "DERECHA"

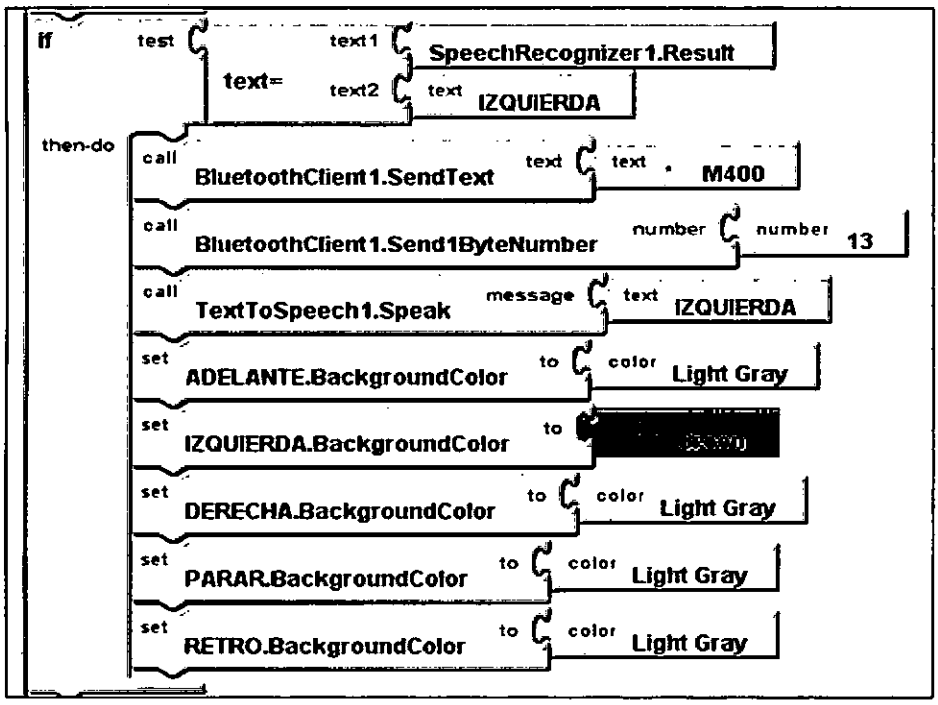


Fig. 1.68 Reconocimiento de Voz, Movimiento "IZQUIERDA"

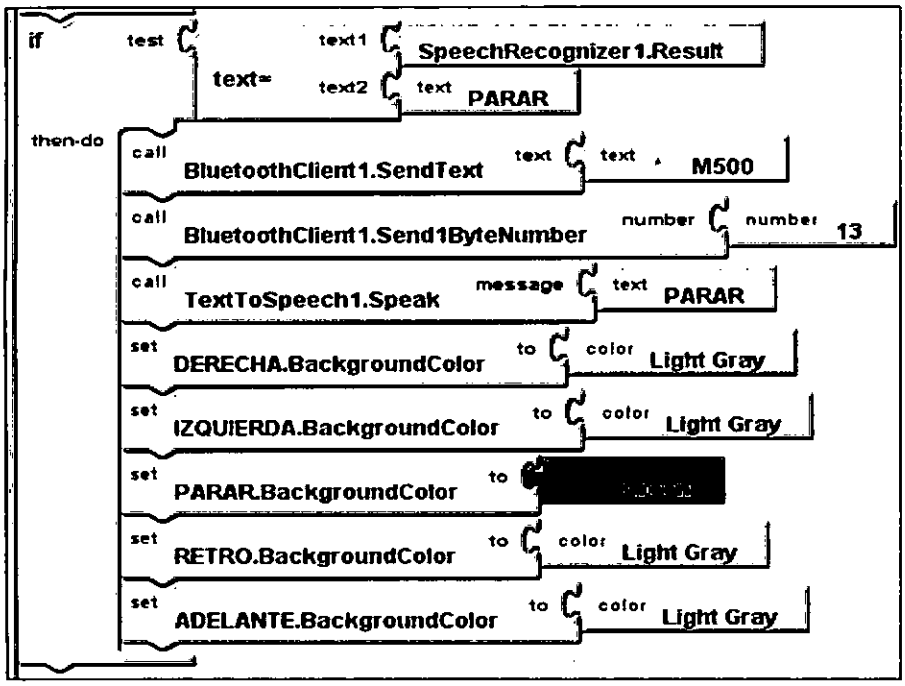


Fig.1.69 Reconocimiento de Voz, Movimiento "PARAR"

4.9. CONTROL PARA VARIACIÓN DE LA VELOCIDAD CON UN CANVAS

Fig.1.70 Cursor con Canvas

Para la variación de la velocidad del motor para dirección hacia "ADELANTE", se utiliza el canvas mediante los eventos Dragged y touched , como se muestra en la Fig.

Con el procedimiento Dragged se determina la velocidad en un porcentaje de 0 y 100%

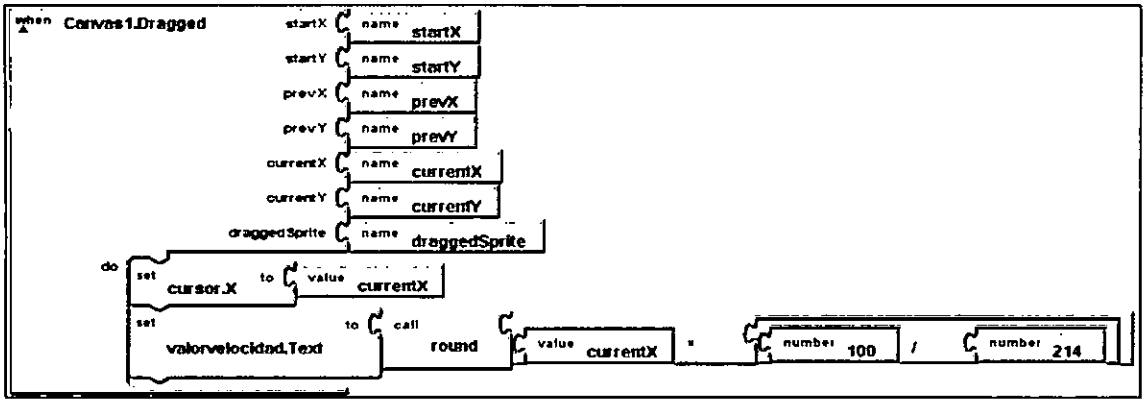


Fig. 1.71 Código determina la posición del curso dentro del canvas

Con el ProcedimientoTouched, una vez determinado el valor entre 0 y 100% al tocar el canvas se envía por Bluetooth el valor.

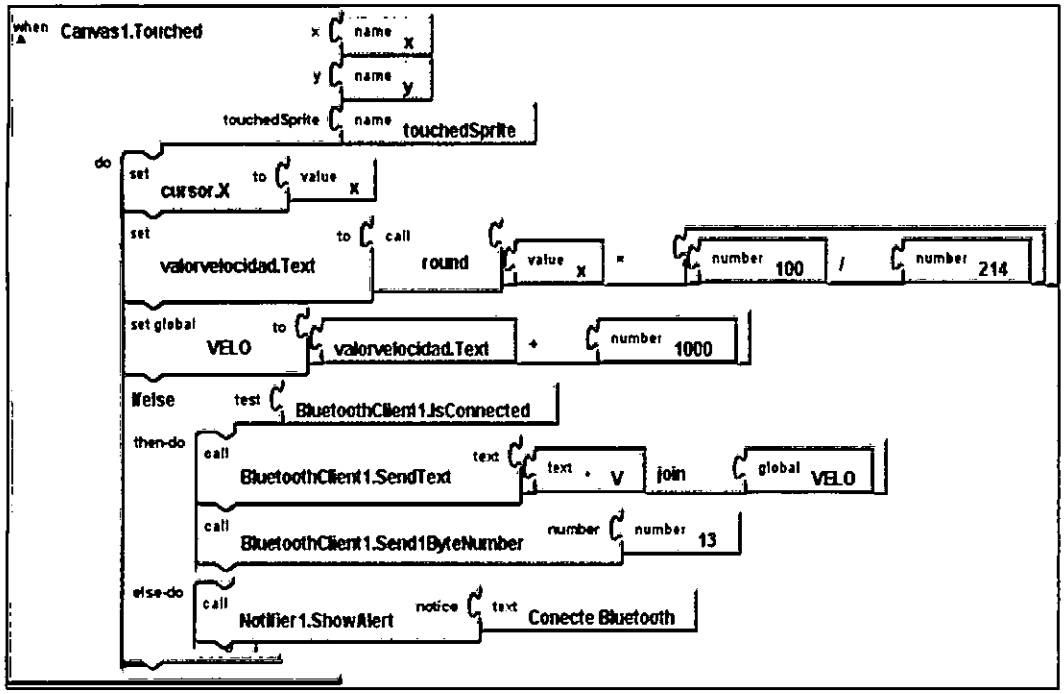


Fig.1.72 Código el valor entre 0 y 100% de la velocidad por bluetooth

4.10. PROGRAMA EN EL MICROCONTROLADOR ATMEGA32

Para programar el microcontrolador se utiliza el compilador BASCOM AVR, y a continuación se mostrara los diagramas de Flujo, en el apéndice se está el código de programa.

4.10.1. DIAGRAMA DE FLUJO PROGRAMA PRINCIPAL

El sistema fue diseñado como sistema redundante, es decir tener varios maneras de control frente alguna falla. Se diseñó con los siguientes controles:

Por Celular:

- Acelerómetro interno del Celular
- Reconocimiento de voz del Celular
- Por Teclado.

Con sensores externos:

- Acelerómetro de la muñeca y cuello
- Tarjeta de Reconocimiento de Voz
- Joystick.

En el siguiente algoritmo se detalla el proceso del programa principal que está cargado en el controlador ATMEGA 32.

El celular comenzará haciendo un barrido de todos los dispositivos bluetooth a su alrededor, hasta encontrar el bluetooth del microprocesador que está conectado por el puerto UART del mismo; después de estar pareados el microntrolador efectuara la siguiente instrucción:

- *Serial0charmatch:*

  Serin Temp , 11 , D , 0 , Mybaud , 0 , 8 , 1

Cuya función es leer la data del puerto serial UART. El comando Serin tiene la función de leer la data del puerto D del micro, esto se debe a que el bluetooth está conectado en los puerto RX y TX, pero solo importa el RX (PIN D0) ya que el microcontrolador se limitara a recibir data que envíe el celular, entonces se almacenará en la variable "Temp", la data serial es tomada desde el bit 1 hasta el bit 8.

Por defecto el programa del controlador viene preseteado a ser utilizado por Teclado de la aplicación hecha, al presionar la tecla de avance, retroceso, izquierda, derecha o parar, se enviara vía bluetooht el carácter "*" seguido de 6 espacios en blanco una letra (M o C), además un número (1-5), seguido de 2 ceros completando así la trama de 11 caracteres por ejemplo (*------M100). Luego el programa saltará a la Subrutina "Controlar" que es la encargada de enviar a las tarjetas de potencia el pulso para efectuar el movimiento de avance-retroceso, izquierda a derecha o parar.

El programa continúa con la subrutina "Velocidad", que controla la rapidez con la que los motores giran. Cuanto mayor sea el Duty Cicle y tienda a la unidad mayor será el pulso PWM y por consiguiente mayor velocidad. Lo que se quería demostrar no era movilizar a una persona más rápido sino de forma segura, es por eso que se decidió que la velocidad se mantendría constante y el usuario seria quien la ajustaría a su necesidad

El microntrolador por defecto está programado para ser utilizado con el teclado del celular pero la siguiente parte del Algoritmo explica que podemos escoger el método con el que podemos controlar nuestra silla. Tenemos 5 formas de control, el código está diseñado para leer los pines en los que están conectados los sensores externos.

Al escoger el Acelerómetro de Mano por ejemplo el programa identificara que fue seleccionado y efectuara la lectura de las entradas en las que está conectado este sensor y comparará con los rangos de valores que fueron obtenidos en la calibración de este sensor para identificar el avance, retroceso, izquierda, derecha.

Si es que se escogiera el reconocimiento de voz por módulo externo, que previamente fue grabado con los comandos para efectuará el movimiento. Este módulo cuenta con 5 pines que están conectados a las I/O digitales del controlador puerto D y parte del B. Cada pin al ser activado efectuara una función diferente, por ejemplo si el primer comando es gradado como AVANZA quedara almacenado en la memoria de la tarjeta de voz en la primera posición cuya salida será el Pin1, entonces cada comando que es almacenado está relacionado a los pines y depende del orden en que fueron grabados para activar un pin respectivo.

Entonces el programa nos pedirá que pronunciemos un comando, la tarjeta de voz procesara la señal y comparará con las de su memoria e identificará el comando para activar los pines correspondientes. Se grabó como primer comando AVANZA que activara el Pin1, RETRO que activara el Pin2, IZQUIERDA que activara el Pin3, DERECHA que activara el Pin4, PARAR que activara el Pin5 de la tarjeta de Voz.

Al identificar cualquiera de los comandos de voz almacenados, lo siguiente que efectuará el programa será ir a la Subrutina "Controlar1" que difiere a la subrutina "Controlar" en que utilizando sensores externos no enviaremos data a través del bluetooth sino trabajaremos directamente con microcontrolador y eliminamos los milisegundos teniendo respuestas más rápidas. Cada vez que la tarjeta de voz reconoce un comando y activa una salida para el giro de uno de los motores, la tarjeta será reseteada y pedirá inmediatamente un nuevo comando de voz y no habrá un retardo a diferencia de la subrutina "Controlar".

Como vemos el programa tiene más de una manera de efectuar el control de la silla. El siguiente control puede ser efectuado por el Joystick; esta tarjeta tiene dos entradas análogas del microcontralador. Al seleccionar el control mediante "Joystick" el programa tiene escrito parámetros que fueron obtenidos en la calibración y que serán detallados más adelante. El programa luego ira a la Subrutina "Controlar1" que fue detallada líneas arriba.

Otra forma de control es el control por "Acelerometro en el cuello" y de manera similar al "Acelerometro en Muñeca" cuenta con parámetros que fueron obtenidos al momento de la calibración, que serán explicados más adelante, pero la lógica es la misma se efectuara el salto a la Subrutina "Controlar1" y realizara el control de los motores.

Además en programa, tenemos 3 casos más que son los que nos facilitaron la calibración de los 2 Acelerómetros y el Joystick, nos proporcionaron el rango de valores que había que considerar para efectuar un movimiento hacia adelante, atrás, izquierda, derecha o para, más adelante se describirán a detalle su funcionamiento.

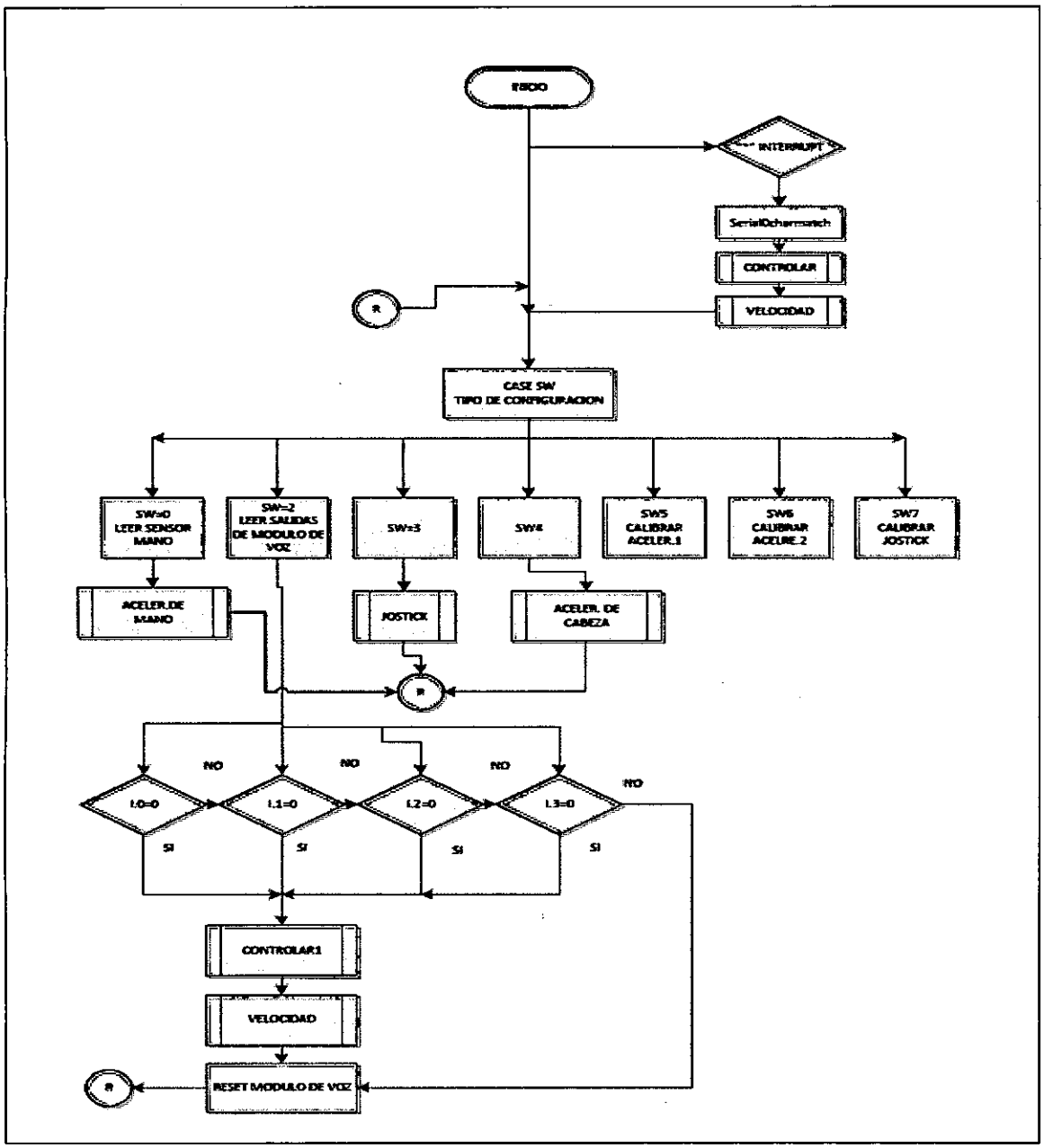


Fig.1.73 Algoritmo del Programa Principal del Microcontrolador.

4.10.2. DIAGRAMA DE FLUJO DEL SUBPROGRAMA JOYSTICK

Este algoritmo explica, previa selección del control de la silla por medio del Joystick.

En esta parte del programa se establecen rango de valores obtenidos en la calibración del Joystick. Se obtuvieron tanto valores con respecto al eje X que determinó el movimiento de Avance-Retroceso; valores con respecto al eje Y que determinó movimiento de Izquierda-Derecha, además del rango de Parada.

A continuación se describe; si el rango de Vx está comprendido entre 124 y 135, además si Vx es mayor igual que 128 y Vy es menor igual que 136 la silla se detendrá, caso contrario si Vy está entre 0 y 70 realizará un movimiento hacia la derecha finalmente si Vy está entre 200 y 255 realizará movimiento hacia la izquierda.

De otro lado con respecto al Eje Y, si Vy está comprendido entre 124 y 135 además si Vx está entre 140 y 255 la silla Retrocederá, caso contrario si Vx está entre los valores 0 y 40 Avanzará.

El microcontrolador al conocer el valor tanto de Vx o Vy, continuara a la subrutina "Controlar1", que líneas arriba explicamos, encargada del movimiento de los motores cuando utilizamos Sensores Externos, en este caso el joystick.

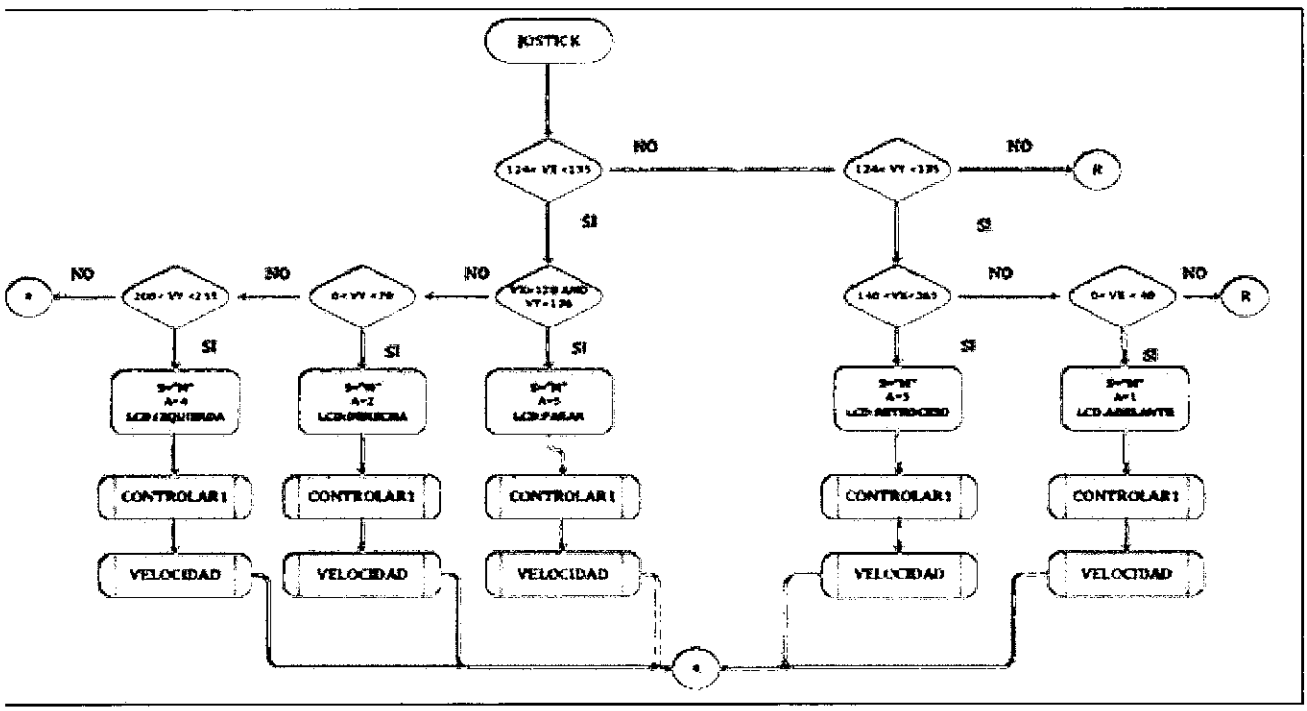


Fig.1.74 Algoritmo del Subprograma del Joystick.

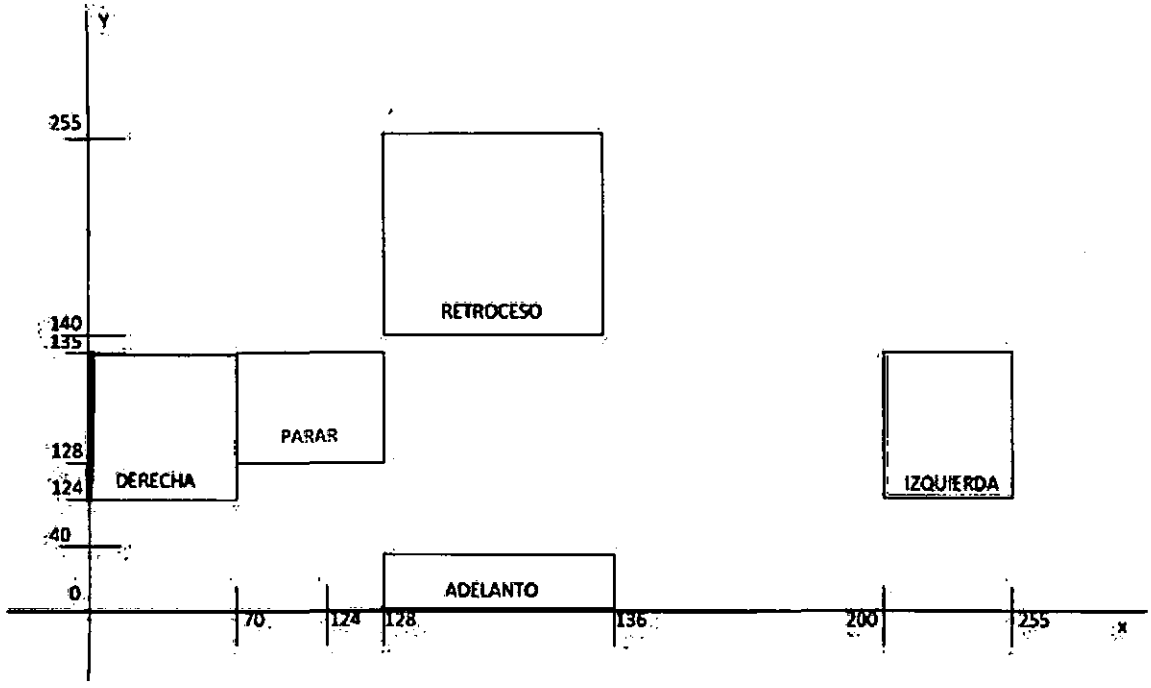


Fig. 1.75 Rango de Valores del pre-escalamiento del Joystick

4.10.3 DIAGRAMA DE FLUJO DEL SUBPROGRAMA PARA CONTROL DE SILLA CON LA MANO. (ACELERÓMETRO MANO)

Este algoritmo explica, que cuando se selecciona como método de control el Acelerómetro de mano, este sensor cuenta con 3 salidas analógicas, uno con respecto al eje X, otra con Y, y finalmente Z, luego ingresan a los puertos ADC del microcontrolador para ser tratadas.

Todos los valores que aparecen a continuación fueron obtenidos previamente al momento de la calibración del sensor, es decir si el acelerómetro era colocada de forma que podía interpretarse como Avance los valores que aparecían en el LCD, que eran valores que habían sido tratados por ADC interno del micro, iban siendo recopilados para luego formar un rango de valores. De la misma manera se trataron las diferentes posiciones como Parar, Retroceso, Izquierda, Derecha.

An2 es el movimiento con respecto al eje Y, An1 es el movimiento con respecto al eje X; entonces en programa se tiene que, si An2 es mayor igual que 39 y An2 menor igual que 45 además si An1 es mayor igual que 39 y An1 menor igual que 40 el sistema saltara a a Subrutina "Control1" para efectuar la orden a los motores.

Si An2 es mayor igual que 39 y An2 menor igual que 45 además si An1 es mayor igual que 42 y An1 menor igual que 47, saltara la Subrutina "Control1" y activará los correspondientes motores; el sistema irá hacía la Izquierda.

La lógica descrita líneas arriba es la misma para el resto de movimientos, lo importante es tener la tabla de valores para el movimiento.

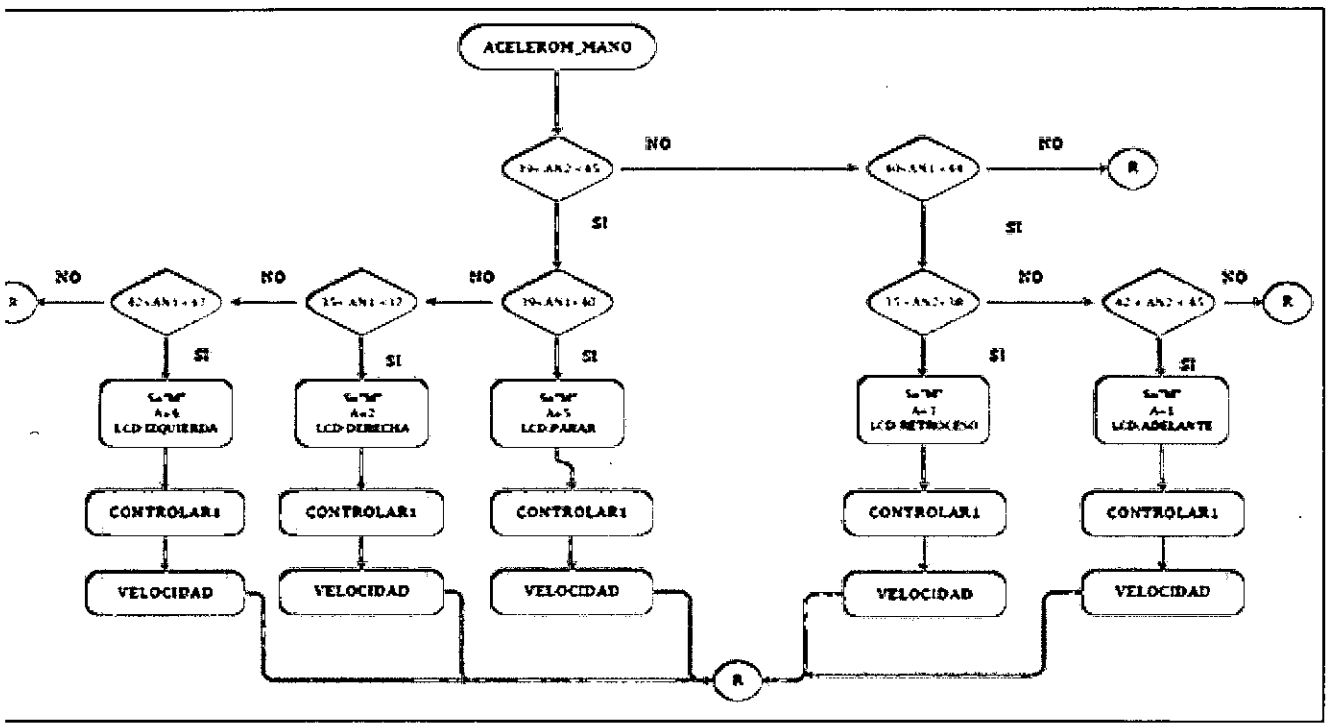


Fig. 1.76 Diagrama de Flujo para el Acelerómetro de Mano.

4.10.4 DIAGRAMA DE FLUJO DEL SUBPROGRAMA PARA CONTROL DE SILLA CON LA CABEZA. (ACELERÓMETRO DE CUELLO)

Funciona de la misma manera que el Programa de Acelerómetro de Mano pero la diferencia radica en el rango de valores debido a que el movimiento del cuello es menor que el movimiento de la mano,

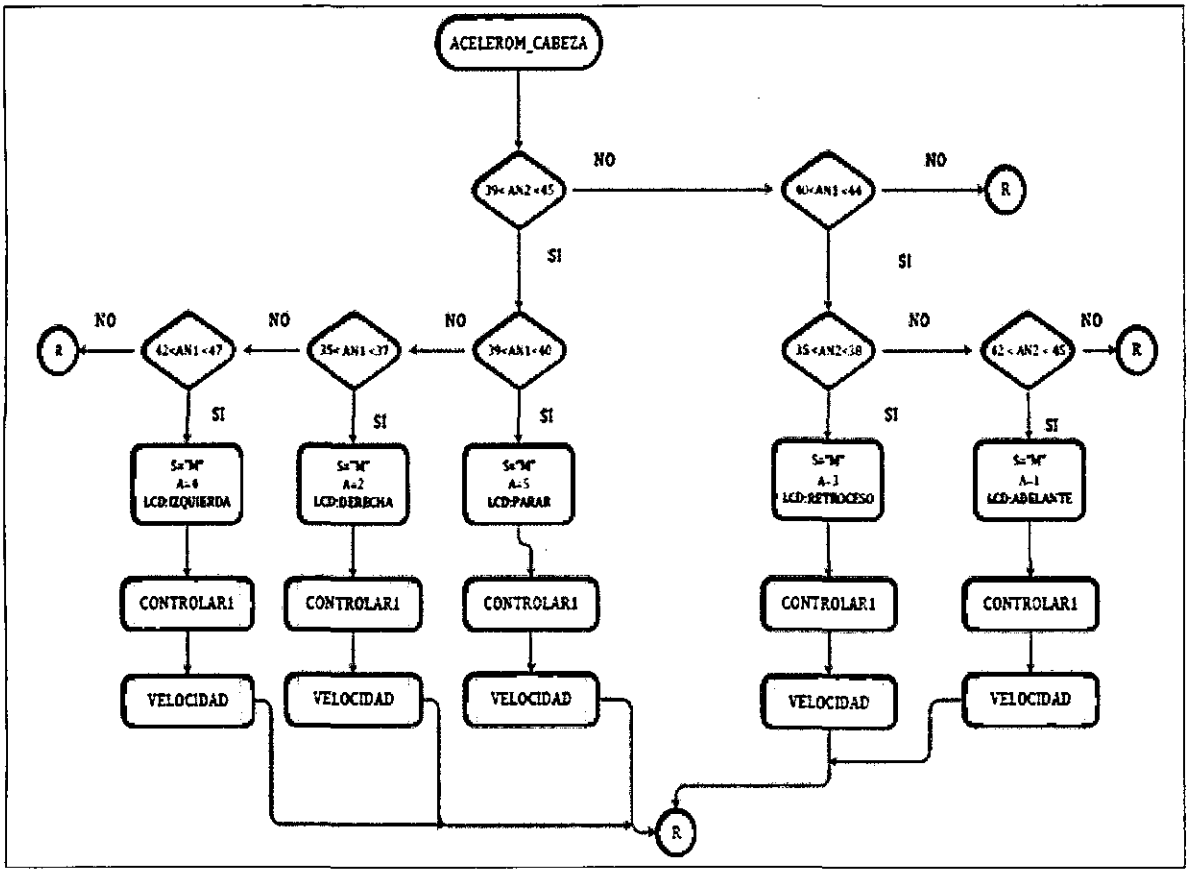


Fig.1.77 Diagrama de Flujo para el Acelerómetro de Cuello.

4.10.5. DIAGRAMA DE FLUJO PROGRAMA DE INTERRUPCIÓN (SERIAL0CHARMATCH)

Como explicáramos líneas arriba, el comando Serial0charmatch es utilizado para activar el canal 0 del puerto Serial UART (Universal Asynchronous Receiver-Transmitter).

El Atmega 32 está preparado, el comando Serin tiene la función de leer la data del puerto D del micro, esto se debe a que el bluetooth está conectado en los puerto RX y TX, pero solo importa el RX (PIN D0) ya que el microcontrolador se limitara a recibir data que envíe el celular, entonces se almacenará en la variable "Temp", la data serial es tomada desde el bit 1 hasta el bit 8.

Por defecto el programa del controlador viene presteado para ser utilizado por Teclado de la aplicación hecha, al presionar la tecla de avance, retroceso, izquierda, derecha o para, se enviara vía bluetooht el carácter "*" seguido de 6 espacios en blanco, seguido de una letra (M o V) y finalmente un número (1-5) seguido de 2 ceros completando así la trama de 11 caracteres, por ejemplo (*------M100). Luego el programa saltará a la Subrutina "Controlar" que es la encargada de enviar a las tarjetas de control y potencia el pulso para efectuar el movimiento de avance-retroceso, izquierda a derecha o parar.

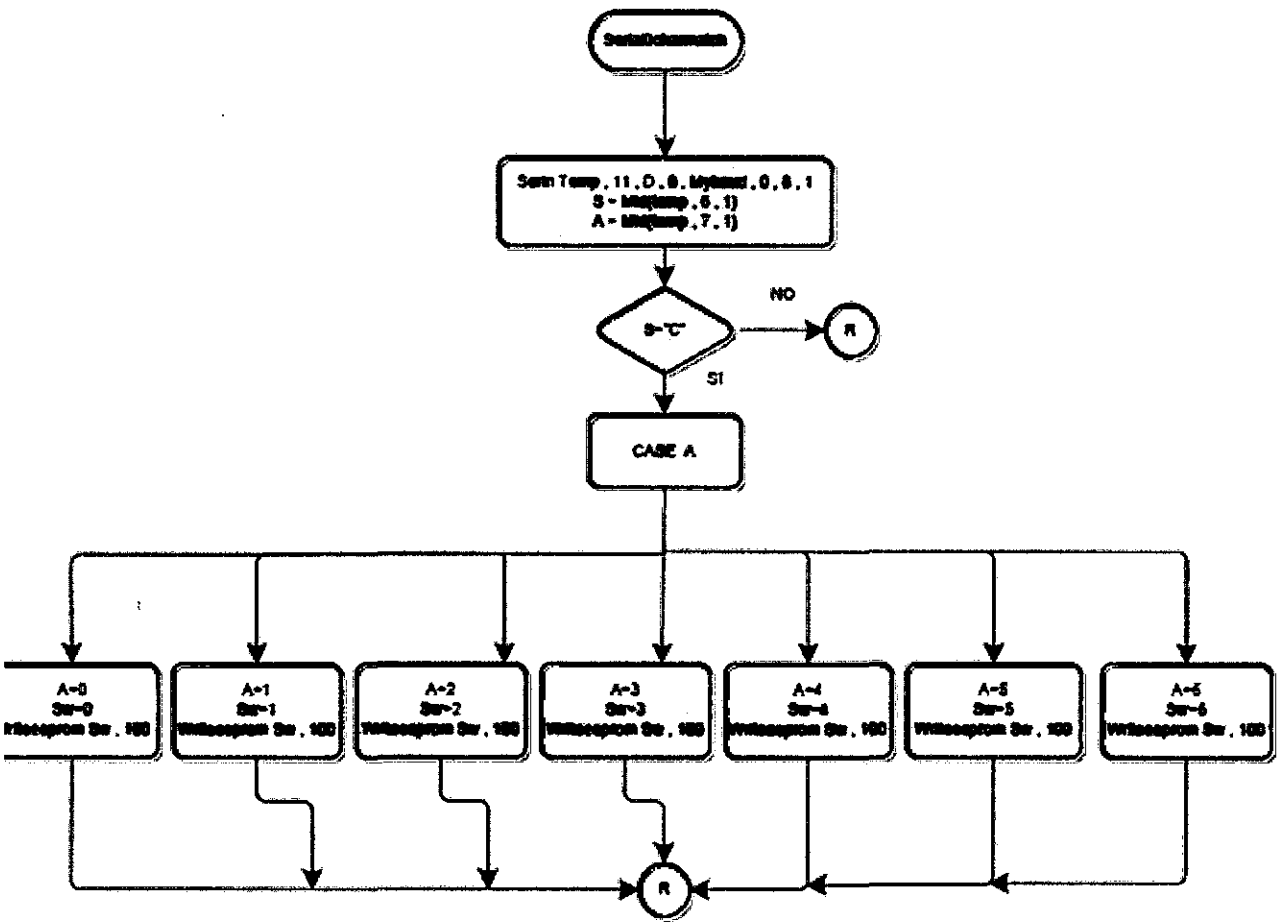


Fig. 1.78 Diagrama de Flujo de Seria0charmatch

4.10.6. DIAGRAMA DE FLUJO DEL SUBPROGRAMA CONTROLAR

El subprograma "Controlar" espera recibir una trama de 11 bits, que fue explicado líneas arriba por ejemplo "*------M100".

Espera recibir la letra "M" o "V", en el caso de recibir M preguntará por el Caso ya sea 1, 2, 3, 4 o 5. Cada caso ejecutará un movimiento diferente que ya han sido detallados anteriormente.

Esta Subrutina "Controlar" también te la opción de que por teclado podamos regular la velocidad del motor mediante PWM. Siempre que se utilice los sensores propios del celular se utilizará está subrutina debido a que está estrechamente ligada a los bluetooths.

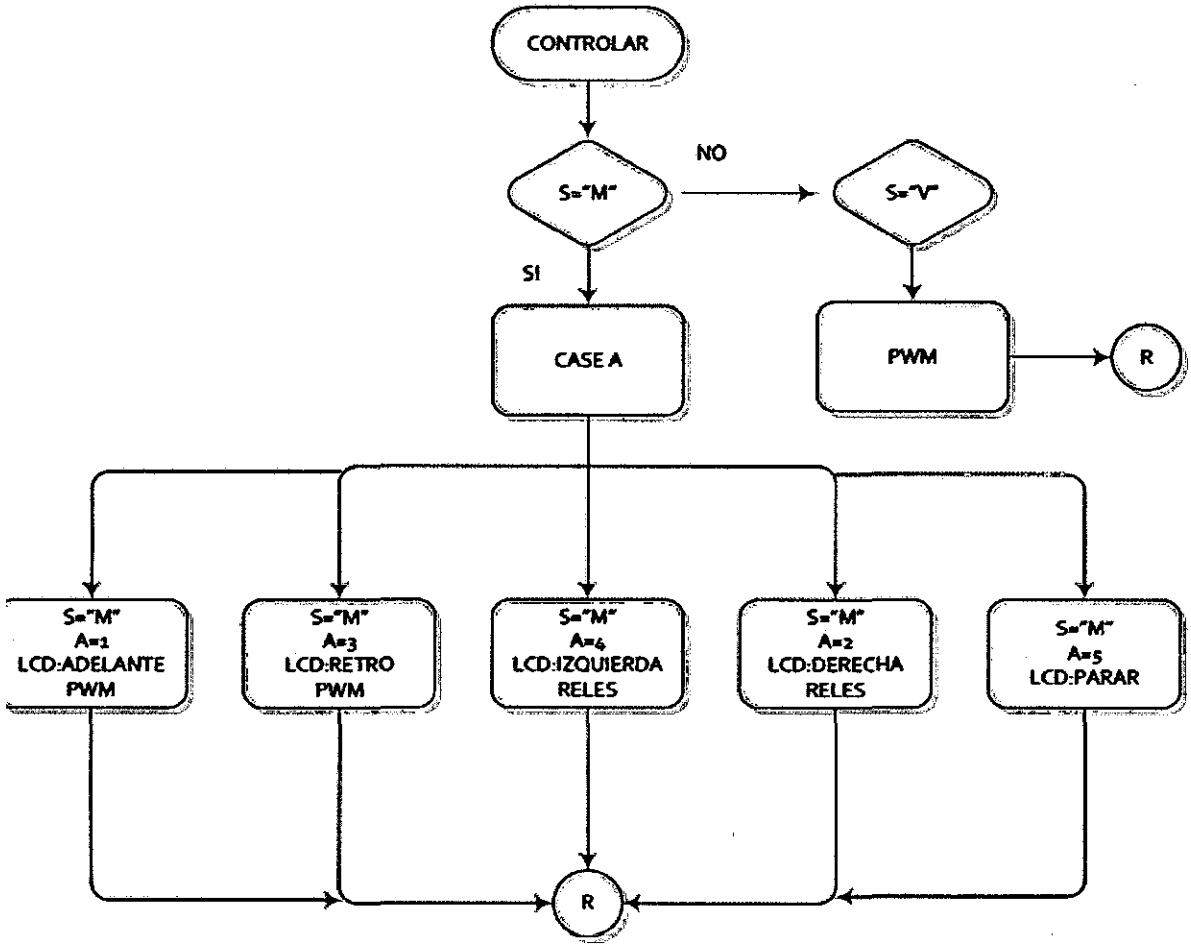


Fig. 1.79 Diagrama de Flujo del subprograma Controlar.

4.10.7. DIAGRAMA DE FLUJO DEL SUBPROGRAMA CONTROLAR1

El subprograma "Controlar1" realiza la misma función que "Controlar" pero la diferencia radica que el programa está estrechamente ligada al control son sensores externos, además no se pregunta por la velocidad con la que deben ir lo motores, sino directamente solicita el caso para efectuar los movimientos que están almacenados en memoria y que fueron descritos.

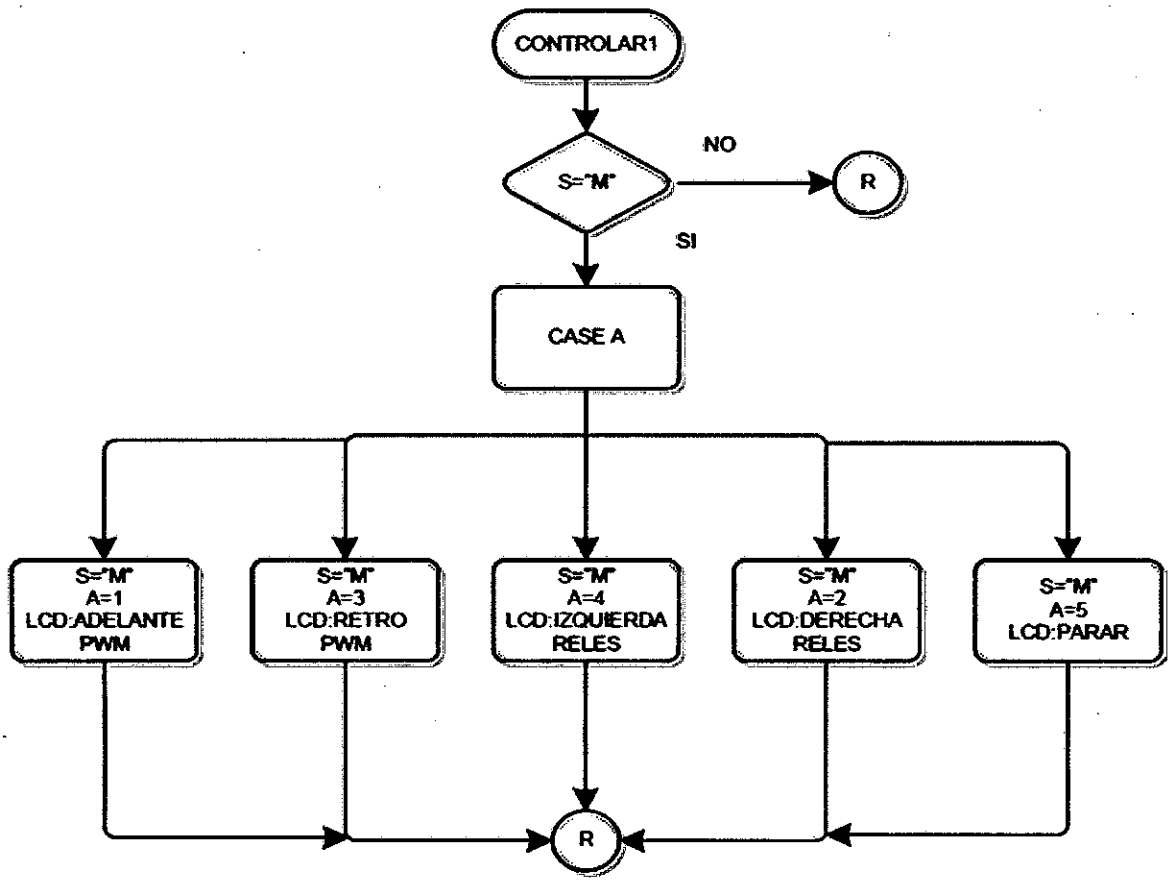


Fig. 1.80 Diagrama de Flujo del subprograma Controlar1.

CAPITULO V

5. RESULTADOS FINALES

En este capítulo se mostrará el aspecto final de la silla de ruedas y sus diferentes componentes para esto se usar fotografías, y videos los cuales también se encuentran ene l CD de la tesis, así también se recomendara algunos cambios para el mejor funcionamiento de esta.

5.3.1. COMPONENTES Y ESPECIFICACIONES TÉCNICAS

- Silla de ruedas común
- Motor DC 12V,180W -Principal
- Motor DC 12V, 120W. Giro
- Microcontrolador Atmega32.
- Sensor de Movimiento Acelerómetro para mano.
- Sensor de Movimiento Acelerómetro para cabeza
- Módulo de reconocimiento de VOZ
- Módulo de Comunicación Bluetooth
- Módulo de Potencia PWM con Mosfets para Motor Principal
- Módulo de potencia con relés para motor de giro
- Equipo celular o Móvil con S.O Android, Bluetooth, Acelerómetro.

5.3.2. FUNCIONAMIENTO

Con el Celular se configura la opción con la que se va ha controlar el movimiento de la silla, las cuales pueden ser:

1. Control con el celular (por botones, Voz, acelerómetro),

2. Movimiento de la mano,

3. Movimiento de la cabeza

4. Por voz

5. Modo manual con Joystick.

5.4. RECOMENDACIONES

Se recomienda utilizar dos motores Brush Motor DC de 24V, 250W, y dos driver tipo puente H de potencia basados en los Mosfets BTS7960B ha 24V DC Ver Fig., los códigos de los programas de microcontrolador y equipos móviles no varian

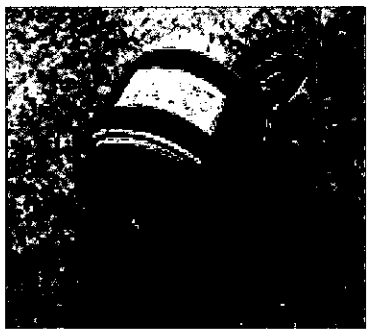


Fig. 1.81 Brush Motor DC 24V, 250W

Fig. 1.82 Max 43A BTS7960B, Driver Para control de Motor DC por PWM

CAPTIULO VI

6. COSTOS DEL PROYECTO

Los costos del proyecto de muestra en la Tabla 6.1

COMPONENTES ELECTRONICOS	CANTIDAD	P.SOLES	TOTAL SOLES	TOTAL DOLLARES
MICROCONTROLADOR	1	40	S/. 40.00	USD 14.18
Acelerómetro	2	35	S/. 70.00	USD 24.82
Módulo Bluetooth	1	50	S/. 50.00	USD 17.73
Módulo de Potencia Mosfets	1	135	S/. 135.00	USD 47.87
Módulo de Potencia Relés	1	35	S/. 35.00	USD 12.41
Módulo de reconocimiento de Voz	1	115	S/. 115.00	USD 40.78
Módulo Joystick	1	40	S/. 40.00	USD 14.18
Construcción de Tarjeta Electrónica	1	50	S/. 50.00	USD 17.73
Tablet -Android	1	300	S/. 300.00	USD 106.38
Caja plástica para tarjeta	1	30	S/. 30.00	USD 10.64
Componentes Mecánicos				
Silla de Ruedas Normal	1	420	S/. 420.00	USD 148.94
Sistema de transmisión	1	150	S/. 150.00	USD 53.19
Motor Principal	1	400	S/. 400.00	USD 141.84
Motor de Dirección	1	280	S/. 280.00	USD 99.29
Batería	1	193	S/. 193.00	USD 68.44
			S/. 2,308.00	USD 818.44
Otros 10%			S/. 230.80	USD 81.84
Total			S/. 2,538.80	USD 900.28

Tabla 1.8 Costos del Proyecto de Muestra

Se realizó la investigación de precios de Sillas Eléctricas en el Perú y los costos varía desde unos miles de dólares, el costo de nuestro propuesta es de S/. 2538 nuevos soles lo que la hace más asequible para más personas.

VII CONCLUSIONES Y RECOMENDACIONES

- Se logró diseñar e Implementar un Sistema de control de una Silla ruedas que permita ser controlada por una persona parapléjica con el mínimo esfuerzo usando el movimiento traslacional y/o voz.

- Se logró Implementar el sistema de comunicación entre la silla de ruedas y el usuario usando la Tecnología Bluetooth y Teléfonos Celulares con Sistema Operativo Android

- Se implementó el hardware electrónico para control de la Silla, para que se controle, por bluetooth, voz y movimiento traslacional mediante acelerómetros.

- Se desarrolló aplicaciones en plataforma Android, para tabletas y Celular que permita controlar la silla y sistema de Ubicación por GPS.

- Se logro adaptar sistema de transmisión Mecánica para una silla de ruedas normal.

- Se determina que la potencia que se logra es de 180W, lo suficiente para transportar una persona de uno 50 kilos o un niño.

- Se puede mejorar la potencia, mejorando sistema de transmisión mecánica, aumentando potencia de motores y amentando capacidad de manejo de corriente del módulo de potencia que maneja los motores., la lógica del programa del microcontrolador y el sistema Android no cambiaría.

VIII BIBLIOGRAFIA

1. INTRODUCCIÓN A LOS MICROCONTROLADORES - José A., González Vásquez.
2. MICROCRONTROLADORES PIC - J. M. Angulo Usastegui, E. Martín Cuenca Angulo Martínez.
3. PIC'N UP THE PACE PIC 16/17 MICROCONTROLLER APPLICATIONS GUIDE, David Benson.
4. NATIONAL DATA BOOK, DALLAS DATA BOOK.
5. ENCICLOPEDIA DE MICROSOFT VISUAL BASIC 2008 - Francisco Javier Cevallos.
6. Curso Programación Android v2,2011, Salvador Gómez Oliver, Auto Edición
7. Guía de desarrollo de aplicaciones para Smartphones y Tabletas, 2013, Sébastian PÉROCHON, Editorial ENI.
8. Crea tus propias aplicaciones Android con Google App Inventor,2012, David Wolber (Autor), Hal Abelson (Autor), Ellen Spertus(Autor), Liz Looney (Autor),Editorial ANAYA.
9. Articulo Apuntes para un aprendiz de programador: App Inventor, programación en dispositivos móviles al alcance de todos. http://www.utm.mx/edi_anteriores/temas45/2NOTAS_45_3.pdf.
10. Articulo "App Inventor; Create Your Own Android Apps App Inventor; Create Your Own Android Apps". http://cs.usfca.edu/~wolber/appinventor/bookSplits/ch2PaintPot.pdf
11. Artículo: "Aplicaciones de un acelerómetro para la medición de inclinaciones horizontales y verticales utilizando microcontroladores avanzados y comunicación serial datalogger e interfaz gráfica. Fuente de energía 4 pilas recargables".

APENDICES

Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

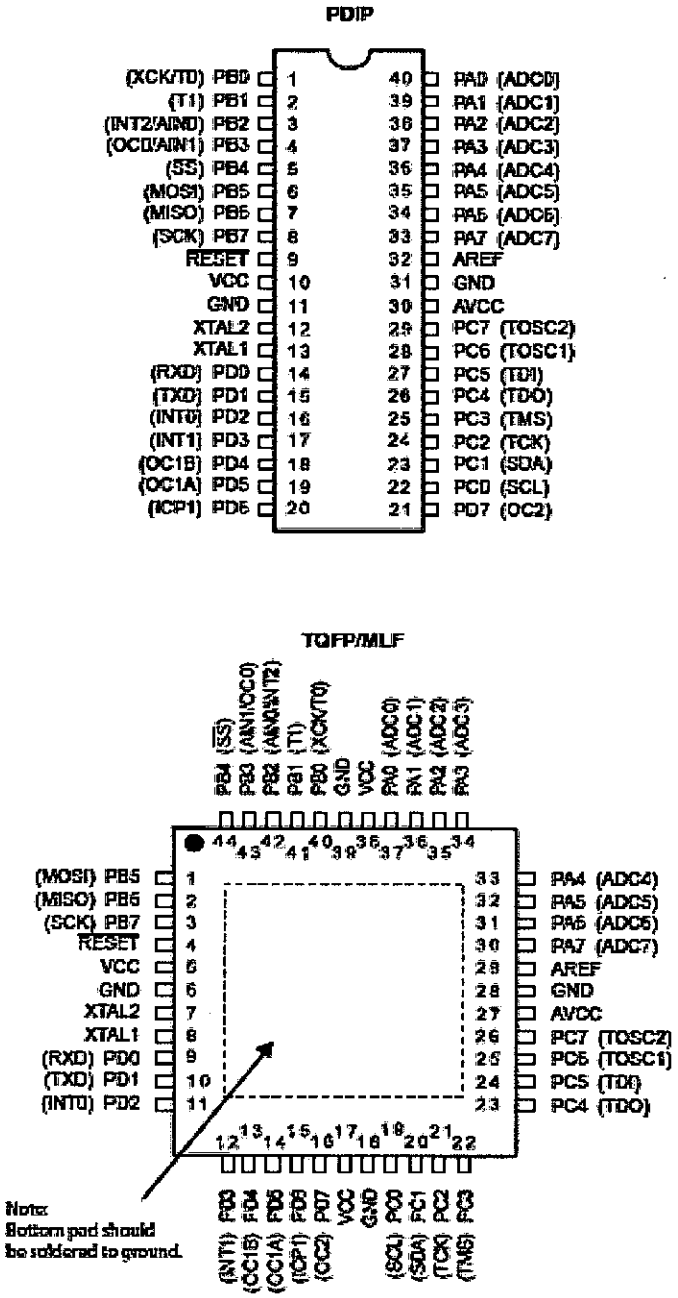
8-bit AVR® Microcontroller with 32KBytes In-System Programmable Flash

ATmega32
ATmega32L

2503Q–AVR–08/11

Pin Configurations

Figure 1. Pinout ATmega32

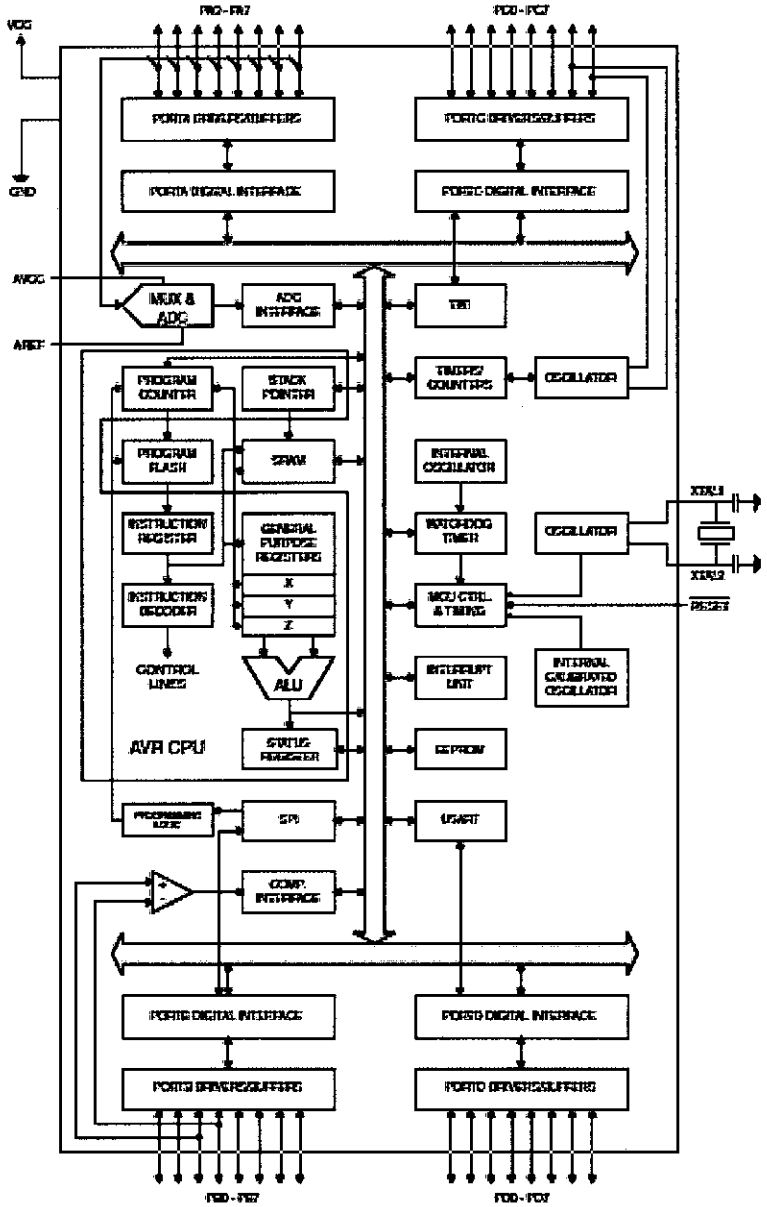


Overview

The Atmel® AVR® ATmega32 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The Atmel® AVR® AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega32 provides the following features: 32Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 1024bytes EEPROM, 2Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega32 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The Atmel AVR ATmega32 is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

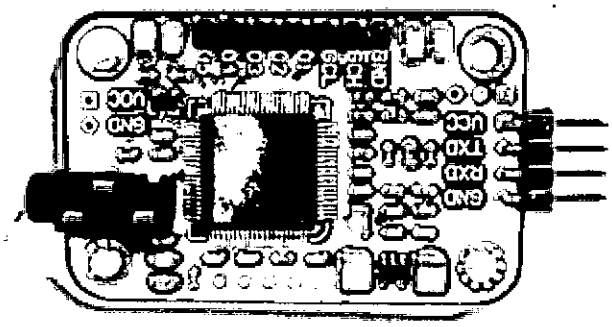
GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Voice Recognition Module V2

Speak to control (Arduino Compatible)



Introduction

The module could recognize your voice. It receives configuration commands or responds through serial port interface. With this module, we can control the car or other electrical devices by voice.

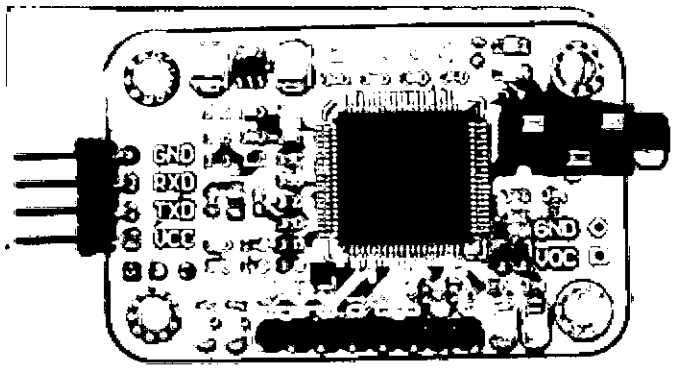
This module can store up to 15 pieces of voice instruction. Those 15 pieces are divided into 3 groups, with 5 in each group. First we should train the module with voice instructions group by group. After that, we should import one group before it could recognize the 5 voice instructions within that group. If we need to implement instructions in other groups, we should import the group first. This module is speaker dependent. If you trained the module, your friend might not be able to make it work.

What's new in V2

We've updated this module to V2. We made V2 easy to control. Except only serial input or output of V1, V2 has other useful ways to control and output the result.

You could find a new GOPI row on V2. GCH and GCL are used to import the voice group. And O1~O5 are pins which output the result of voice recognition. For example, if the first voice instruction in the working group is recognized, O1 could output HIGH signal. This output sometimes is very useful, such as while controlling the relay.

The O1~O5 output could be set as many type. You could set it by sending command to it through serial interface. Those setting will be recorded in memory. It will not lose even with power off. You could find the commands in later content.



V1 is black and V2 is red.

Technical

Parameters

- *Voltage: 4.5-5.5V*
- *Current: <40mA*
- *Digital interface: 5V TTL level UART interface and GPIO*
- *Analog interface: 3.5mm mono-channel microphone connector + microphone pin interface*
- *Size: 30mm x 47.5mm*
- *Recognition accuracy: 99% (under ideal environment)*

Serial Command

This module can be configured by sending commands via serial port. Configuration will be not erased after powered off.

Its interface is 5V TTL. The serial data format: 8 data bits, no parity, 1 stop bit. The default baud rate is 9600 and baud rate can be changed.

Command format is "Head + Key". "Head" is a 0xaa, and "Key" is as follows:

Key (HEX format)	Description	Respond in Common Mode	Respond in Compact Mode
0x00	Enter into "Waiting" state	"Waiting! \ n" : successful "ERROR! \ n" : instruction error	0xcc : successful 0xe0 : instruction error
0x01	Delete the instructions of group 1	"Group1 Deleted ! \ n" : successful "ERROR! \ n" : instruction error	0xcc : successful 0xe0 : instruction error
0x02	Delete the instructions of group 2	"Group2 Deleted ! \ n" : successful "ERROR! \ n" : instruction error	0xcc : successful 0xe0 : instruction error
0x03	Delete the instructions of group 3	"Group3 Deleted ! \ n" : successful "ERROR! \ n" : instruction error	0xcc : successful 0xe0 : instruction error
0x04	Delete the instructions of all the 3 groups	" All Groups Deleted !\n " : successful "ERROR! \ n" : instruction error	0xcc : successful 0xe0 : instruction error
0x11	Begin to record instructions of group 1	"ERROR! \ n" : instruction error "START \ n" : Ready for recording, you can speak	0xe0 : instruction error 0x40 : Ready for recording, you

		now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group1 finished! \ n" : finish recording group 1	can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x46 : finish recording group 1
0x12	Begin to record instructions of group 2	"ERROR! \ n" : Instruction error "START \ n" : Ready for recording, you can speak now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group2 finished! \ n" : finish recording group 2	0xE0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x47 : finish recording group 2
0x13	Begin to record instructions of group 3	"ERROR! \ n" : Instruction error "START \ n" : Ready for recording, you can speak now "No voice \ n" : no voice detected "Again \ n" : Speak the voice instruction again. Do not speak until getting the START message "Too loud \ n" : Too loud to record "Different \ n" : voice instruction confirming failed. Voice for the second chance is different with the first one. "Finish one \ n" : recording one voice instruction successfully "Group3 finished! \ n" : finish recording group 3	0xE0 : Instruction error 0x40 : Ready for recording, you can speak now 0x41 : no voice detected 0x42 : Speak the voice instruction again. Do not speak until getting the START message 0x43 : Too loud to record 0x44 : voice instruction confirming failed. Voice for the second chance is different with the first one. 0x45 : recording one voice instruction successfully 0x48 : finish recording group 3
0x21	Import group 1 and be ready for voice instruction	"Group1 imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xCC : Successful 0xE0 : Instruction error 0xE1 : Importing voice group failed
0x22	Import group 2 and be ready for voice instruction	"Group2 imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xCC : Successful 0xE0 : Instruction error 0xE1 : Importing voice group failed
0x23	Import group 3 and be ready for voice instruction	"Group3 imported !\n" : Successful "ERROR! \ n" : Instruction error "Import failed !\n" : Importing voice group failed	0xCC : Successful 0xE0 : Instruction error 0xE1 : Importing voice group failed

0x24	Query the recorded group	"Used group:0\n" : No group is recorded "Used group:1\n" : Group 1 is recorded "Used group:2\n" : Group 2 is recorded "Used group:3\n" : Group 3 is recorded "Used group:12\n" : Group 1 and Group 2 are recorded "Used group:13\n" : Group 1 and Group 3 are recorded "Used group:23\n" : Group 2 and Group 3 are recorded "Used group:123\n" : All the 3 groups are recorded "ERROR! \ n" : Instruction error	0x00 : No group is recorded 0x01 : Group 1 is recorded 0x02 : Group 2 is recorded 0x04 : Group 3 is recorded 0x03 : Group 1 and Group 2 are recorded 0x05 : Group 1 and Group 3 are recorded 0x06 : Group 2 and Group 3 are recorded 0x07 : All the 3 groups are recorded 0xe0 : Instruction error
0x31	Change the baud rate to 2400bps	"Baud: 2400\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x32	Change the baud rate to 4800bps	"Baud: 4800\n" : Successful "ERROR! \ n" : Instruction error	
0x33	Change the baud rate to 9600bps	"Baud: 9600\n" : Successful "ERROR! \ n" : Instruction error	
0x34	Change the baud rate to 19200bps	"Baud: 19200\n" : Successful "ERROR! \ n" : Instruction error	
0x35	Change the baud rate to 38400bps	"Baud: 38400\n" : Successful "ERROR! \ n" : Instruction error	
0x36	Switch to Common Mode	"Common Mode\n" : Successful "ERROR! \ n" : Instruction error	
0x37	Switch to Compact Mode	"Compact Mode\n" : Successful "ERROR! \ n" : Instruction error	
0x41	Reset output of O1	"Ok\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x42	Reset output of O2		
0x43	Reset output of O3		
0x44	Reset output of O4		
0x45	Reset output of O5		
0x46	Reset output of O1~O5		
0x50	Set O1~O5 to Pulse Mode (negative pulse)	"Ok\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0x51	Set O1~O5 to Flip Mode		
0x52	Set O1~O5 to Down Mode		
0x53	Set O1~O5 to Up Mode		
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68	Set the output duty of O1~O5 in Pulse Mode 0x60 – 10ms 0x61 – 15ms 0x62 – 20ms 0x63 – 25ms 0x64 – 30ms 0x65 – 50ms 0x66 – 60ms 0x67 – 70ms 0x68 – 80ms 0x69 – 90ms 0x6A – 100ms 0x6B – 200ms 0x6C – 300ms	"Ok\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error

0x69	0x6D – 400ms 0x6E – 500ms 0x6F – 1s		
0x6A			
0x6B			
0x6C			
0x6D			
0x6E			
0x6F			
0x70	Reset Serial Port to: 9600 baud rate, 8 data bits, no parity, 1 stop bit	"Ok\n" : Successful "ERROR! \ n" : Instruction error	0xcc : successful 0xe0 : Instruction error
0xbb	Query version information	Version information	No respond

If you want to modify the serial baud rate to 38400, you need to send command: 0xaa35. If successful, it will return "Baud: 38400 \ n"(in Common Mode) or 0xcc (in Compact Mode). The baud rate is set to 38400.

The main difference between Compact Mode and Common Mode is the returning message. Common Mode response is long string but Compact Mode response is a byte. For example, after sending 0xaa04 to delete all the contents of the 3 groups, in Common Mode it will return "All Groups Deleted! \ n", but in Compact Mode it will return a concise bytes such as 0xcc which means a successful operation.

Here we will introduce more about the output of O1~O5:

Pulse Mode: Output is negative pulse. The pulse duration time (pulse duty) could be set by command 0x60 ~ 0x6F

Flip Mode: each time while the module recognizes voice command, it will change the state of the output pin.

Down Mode: The output will become LOW from HIGH once it detects voice command. It will never come back to HIGH again until the module receives output reset command 0x41 ~ 0x46

Up Mode: The output will become HIGH from LOW once it detects voice command. It will never come back to LOW again until the module receives output reset command 0x41 ~ 0x46

There will be an example:

A: Starting point while you import the voice group.

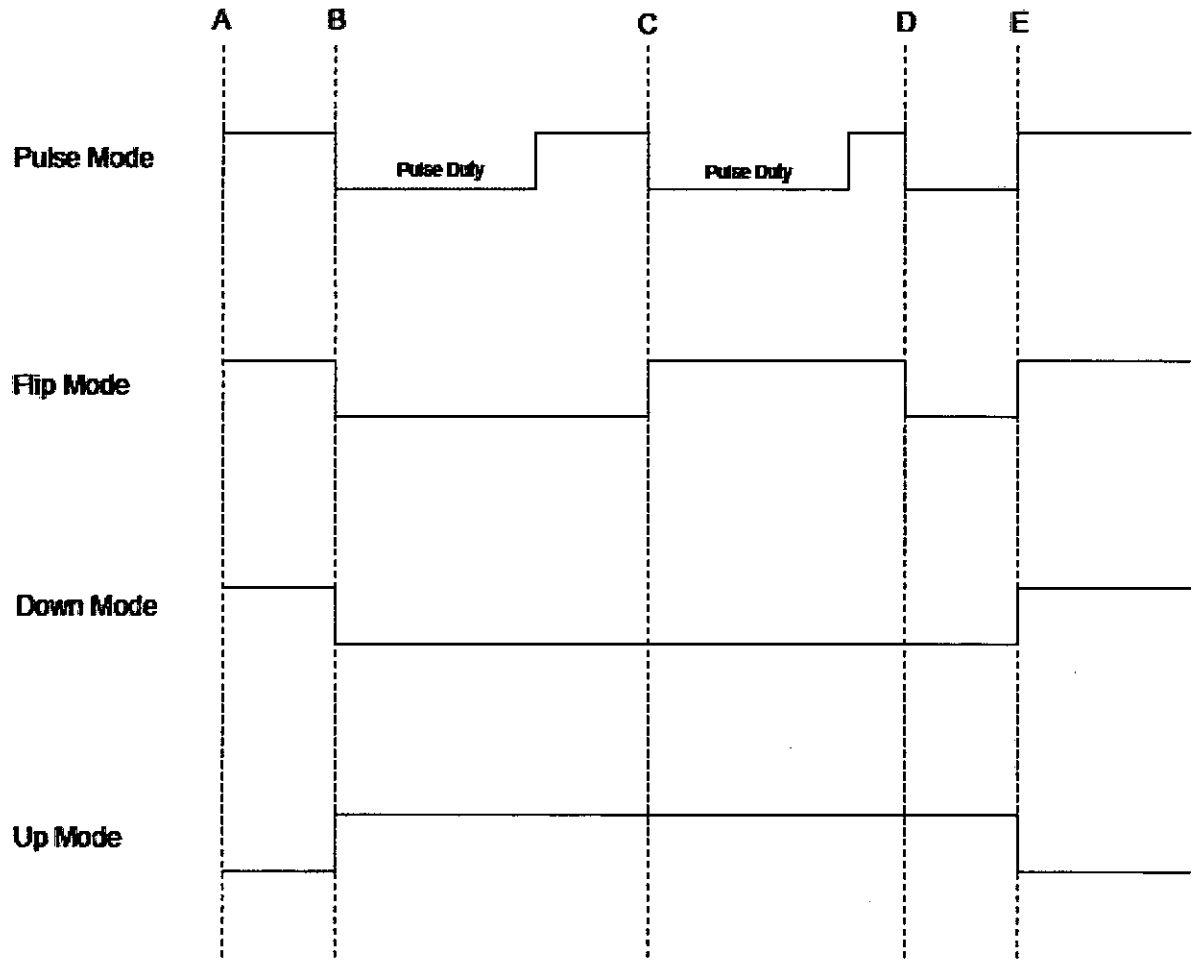
B: The 1st time it recognizes voice command.

C: The 2nd time it recognizes voice command.

D: The 3rd time it recognizes voice command.

E: The time while output reset command is received (0x41 ~ 0x46).

The back wire is output wave shape.



For the first-time use, we need to do some configuration:

1. *Select the serial baud rate (default 9600)*
2. *Select the communication mode: Common Mode or Compact Mode*
3. *Recording five instructions of the first group(or 2nd or 3rd as required)*
4. *Import the group you need to use (only recognize 5 instructions within one group at the same time)*

After all the setting above, you can speak or send voice instruction to it. If identified successfully, result will be returned via serial port in the format: group number + command number. For example, return Result: 11 (Compact mode returns 0x11) means identified the first command of group 1.

If voice instruction is recorded, each time after you power it on, you need to import the group before letting it identify voice instructions.

LED

Recording stage:

1. *Record indication: D1 (RED) flashes 3 times within the 600ms, then off for400ms, and then flashes quickly for 4 times within 600ms. Now the recording indication is over.*
2. *Begin to speak: D1 (RED) is off for 400ms, and then is on. Voice during the time while D1 (RED) is on will be recorded by this module.*
3. *Recording a voice instruction successfully for the first time: D1 (RED) off, D2 (ORANGE) on for 300ms.*
4. *Recording a voice instruction successfully for the first time: D1 (RED) off, D2 (ORANGE) on for 700ms.*
5. *Recording failure: D2 (ORANGE) flashes 4 times within the 600ms. In cases that voice instructions detected twice don't match, or the sound is too large, or there is no sound, recording will fail. You need to start over the recording process for that instruction.*

Waiting mode:

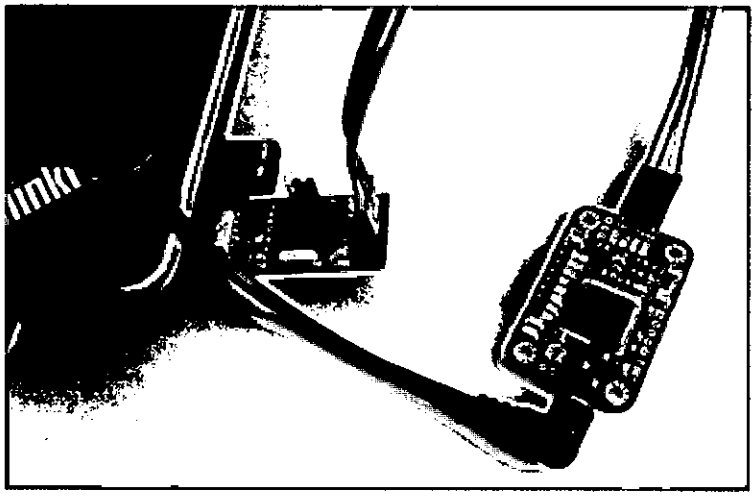
In waiting mode, D2 (ORANGE) is off, and D1 (RED) is on for 80ms every other 200ms, fast flashing. In this mode, it doesn't recognize voice command, only waiting for serial commands.

Recognition stage:

In identification stage, D2 (ORANGE) is off, and D1 (RED) is on for 100ms every other 1500ms, slow flashing. In this stage, this module is processing received voice signal, and if matching, it will send the result immediately via serial port.

Recording

Before using it, we have train it by recording voice instructions. Each voice instruction has the maximum length of 1300ms, which ensures that most words can be recorded. Once you start recording, you can't stop the recording process until you finish all the 5 voice instructions recording of one group. Also, once you start recording, the previous voice instructions in that group will be erased. In training state, this module doesn't reply to any other serial commands.



LED will flash to indicate state. Please refer to the LED part.

First, you need a serial tool. Here we use AccessPort (Download page).

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

- 3-axis sensing
- Small, low profile package
  - 4 mm × 4 mm × 1.45 mm LFCSP
- Low power : 350 µA (typical)
- Single-supply operation: 1.8 V to 3.6 V
- 10,000 *g* shock survival
- Excellent temperature stability
- BW adjustment with a single capacitor per axis
- RoHS/WEEE lead-free compliant

APPLICATIONS

- Cost sensitive, low power, motion- and tilt-sensing applications
  - Mobile devices
  - Gaming systems
  - Disk drive protection
  - Image stabilization
  - Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

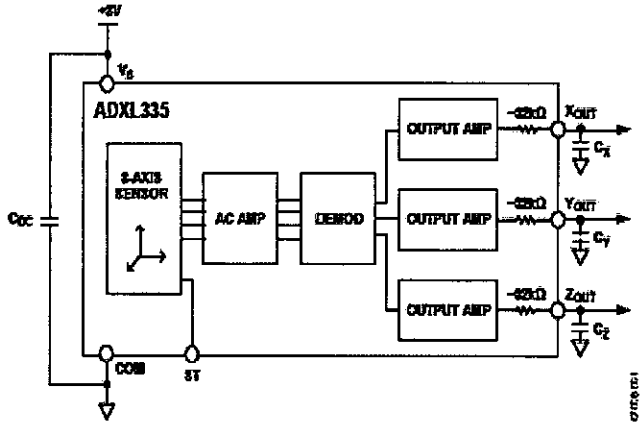


Figure 1.

SPECIFICATIONS

T_A = 25°C, V_S = 3 V, C_X = C_Y = C_Z = 0.1 µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		mg/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µg/√Hz rms
Noise Density Z_{OUT}			300		µg/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × *C*). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered)	10,000 *g*
Acceleration (Any Axis, Powered)	10,000 *g*
Vs	−0.3 V to +3.6 V
All Other Pins	(COM − 0.3 V) to (Vs + 0.3 V)
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Temperature Range (Powered)	−55°C to +125°C
Temperature Range (Storage)	−65°C to +150°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

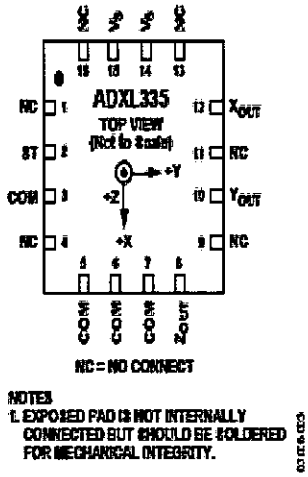


Figure 2. Pin Configuration

Table 3. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	NC	No Connect[1].
2	ST	Self-Test.
3	COM	Common.
4	NC	No Connect[1].
5	COM	Common.
6	COM	Common.
7	COM	Common.
8	Z_{OUT}	Z Channel Output.
9	NC	No Connect[1].
10	Y_{OUT}	Y Channel Output.
11	NC	No Connect[1].
12	X_{OUT}	X Channel Output.
13	NC	No Connect[1].
14	V_S	Supply Voltage (1.8 V to 3.6 V).
15	V_S	Supply Voltage (1.8 V to 3.6 V).
16	NC	No Connect[1].
EP	Exposed Pad	Not internally connected. Solder for mechanical integrity.

[1]NC pins are not internally connected and can be tied to COM pins, unless otherwise noted.

THEORY OF OPERATION

The ADXL335 is a complete 3-axis acceleration measurement system. The ADXL335 has a measurement range of ±3 *g* minimum. It contains a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer can measure the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180° out-of-phase square waves. Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

The demodulator output is amplified and brought off-chip through a 32 kΩ resistor. The user then sets the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

MECHANICAL SENSOR

The ADXL335 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axes' sense directions are highly orthogonal and have little cross-axis sensitivity. Mechanical misalignment of the sensor die to the package is the chief source of cross-axis sensitivity. Mechanical misalignment can, of course, be calibrated out at the system level.

PERFORMANCE

Rather than using additional temperature compensation circuitry, innovative design techniques ensure that high performance is built in to the ADXL335. As a result, there is no quantization error or nonmonotonic behavior, and temperature hysteresis is very low (typically less than 3 mg over the −25°C to +70°C temperature range).

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 µF capacitor, C_{DC}, placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply. However, in applications where noise is present at the 50 kHz internal clock frequency (or any harmonic thereof), additional care in power supply bypassing is required because this noise can cause errors in acceleration measurement.

If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (1 µF or greater) can be added in parallel to C_{DC}. Ensure that the connection from the ADXL335 ground to the power supply ground is low impedance because noise transmitted through ground has a similar effect to noise transmitted through V_S.

SETTING THE BANDWIDTH USING C_X, C_Y, AND C_Z

The ADXL335 has provisions for band limiting the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is

$$F_{-3\,dB} = 1/(2\pi(32\ k\Omega) \times C_{(X,\,Y,\,Z)})$$

or more simply

$$F_{-3\,dB} = 5\ \mu F/C_{(X,\,Y,\,Z)}$$

The tolerance of the internal resistor (R_{FILT}) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 0.0047 µF for C_X, C_Y, and C_Z is recommended in all cases.

Table 4. Filter Capacitor Selection, C_X, C_Y, and C_Z

Bandwidth (Hz)	Capacitor (µF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

SELF-TEST

The ST pin controls the self-test feature. When this pin is set to V_S, an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is −1.08 *g* (corresponding to −325 mV) in the X-axis, +1.08 *g* (or +325 mV) on the Y-axis, and +1.83 *g* (or +550 mV) on the

Never expose the ST pin to voltages greater than V_S + 0.3 V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low V_F clamping diode between ST and V_S is recommended.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The selected accelerometer bandwidth ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor to improve the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at X_{OUT}, Y_{OUT}, and Z_{OUT}.

The output of the ADXL335 has a typical bandwidth of greater than 500 Hz. The user must filter the signal at this point to limit aliasing errors. The analog bandwidth must be no more than half the analog-to-digital sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL335 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of µg/√Hz (the noise is proportional to the square root of the accelerometer bandwidth). The user should limit bandwidth to the lowest frequency needed by the application to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL335 is determined by

$$rms\ Noise = Noise\ Density \times (\sqrt{BW \times 1.6})$$

It is often useful to know the peak value of the noise. Peak-to-peak noise can only be estimated by statistical methods. Table 5 is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table 5. Estimation of Peak-to-Peak Noise

Peak-to-Peak Value	% of Time That Noise Exceeds Nominal Peak-to-Peak Value
2 x rms	32
4 x rms	4.6
6 x rms	0.27
8 x rms	0.006

USE WITH OPERATING VOLTAGES OTHER THAN 3 V

The ADXL335 is tested and specified at V_S = 3 V; however, it can be powered with V_S as low as 1.8 V or as high as 3.6 V. Note that some performance parameters change as the supply voltage is varied.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 µF capacitor, C_{DC}, placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply. However, in applications where noise is present at the 50 kHz internal clock frequency (or any harmonic thereof), additional care in power supply bypassing is required because this noise can cause errors in acceleration measurement.

If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (1 µF or greater) can be added in parallel to C_{DC}. Ensure that the connection from the ADXL335 ground to the power supply ground is low impedance because noise transmitted through ground has a similar effect to noise transmitted through V_S.

SETTING THE BANDWIDTH USING C_X, C_Y, AND C_Z

The ADXL335 has provisions for band limiting the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is

$$F_{-3\,dB} = 1/(2\pi(32\ k\Omega) \times C_{(X,Y,Z)})$$

or more simply

$$F_{-3\,dB} = 5\ \mu F/C_{(X,Y,Z)}$$

The tolerance of the internal resistor (R_{FILT}) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 0.0047 µF for C_X, C_Y, and C_Z is recommended in all cases.

Table 4. Filter Capacitor Selection, C_X, C_Y, and C_Z

Bandwidth (Hz)	Capacitor (µF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

SELF-TEST

The ST pin controls the self-test feature. When this pin is set to V_S, an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is −1.08 *g* (corresponding to −325 mV) in the X-axis, +1.08 *g* (or +325 mV) on the Y-axis, and +1.83 *g* (or +550 mV) on the Z-axis. This ST pin can be left open-circuit or connected to common (COM) in normal use.

Never expose the ST pin to voltages greater than V_S + 0.3 V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low V_F clamping diode between ST and V_S is recommended.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The selected accelerometer bandwidth ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor to improve the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at X_{OUT}, Y_{OUT}, and Z_{OUT}.

The output of the ADXL335 has a typical bandwidth of greater than 500 Hz. The user must filter the signal at this point to limit aliasing errors. The analog bandwidth must be no more than half the analog-to-digital sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL335 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of µ*g*/√Hz (the noise is proportional to the square root of the accelerometer bandwidth). The user should limit bandwidth to the lowest frequency needed by the application to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL335 is determined by

$$rms\ Noise = Noise\ Density \times (\sqrt{BW \times 1.6})$$

It is often useful to know the peak value of the noise. Peak-to-peak noise can only be estimated by statistical methods. Table 5 is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table 5. Estimation of Peak-to-Peak Noise

Peak-to-Peak Value	% of Time That Noise Exceeds Nominal Peak-to-Peak Value
2 × rms	32
4 × rms	4.6
6 × rms	0.27
8 × rms	0.006

USE WITH OPERATING VOLTAGES OTHER THAN 3 V

The ADXL335 is tested and specified at V_S = 3 V; however, it can be powered with V_S as low as 1.8 V or as high as 3.6 V. Note that some performance parameters change as the supply voltage is varied.

The ADXL335 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At V_S = 3.6 V, the output sensitivity is typically 360 mV/*g*. At V_S = 2 V, the output sensitivity is typically 195 mV/*g*.

The zero *g* bias output is also ratiometric, thus the zero *g* output is nominally equal to $V_S/2$ at all supply voltages.

The output noise is not ratiometric but is absolute in volts; therefore, the noise density decreases as the supply voltage increases. This is because the scale factor (mV/*g*) increases while the noise voltage remains constant. At V_S = 3.6 V, the X-axis and Y-axis noise density is typically 120 µg/√Hz, whereas at V_S = 2 V, the X-axis and Y-axis noise density is typically 270 µg/√Hz.

Self-test response in *g* is roughly proportional to the square of the supply voltage. However, when ratiometricity of sensitivity is factored in with supply voltage, the self-test response in volts is roughly proportional to the cube of the supply voltage. For example, at V_S = 3.6 V, the self-test response for the ADXL335 is approximately −560 mV for the X-axis, +560 mV for the Y-axis, and +950 mV for the Z-axis.

At V_S = 2 V, the self-test response is approximately −96 mV for the X-axis, +96 mV for the Y-axis, and −163 mV for the Z-axis.

The supply current decreases as the supply voltage decreases. Typical current consumption at V_S = 3.6 V is 375 µA, and typical current consumption at V_S = 2 V is 200 µA.

AXES OF ACCELERATION SENSITIVITY

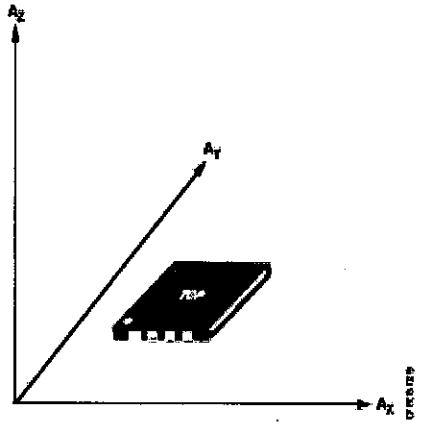


Figure 23. Axes of Acceleration Sensitivity; Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis.

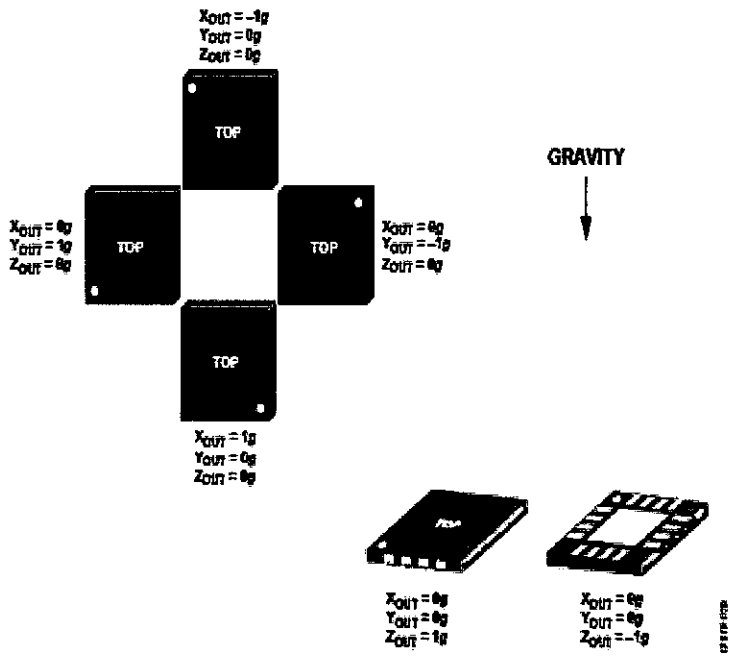


Figure 24. Output Response vs. Orientation to Gravity

BTS 7960

High Current PN Half Bridge
NovalithIC™
43 A, 7 mΩ + 9 mΩ

High Current PN Half Bridge
NovalithIC™

BTS 7960B

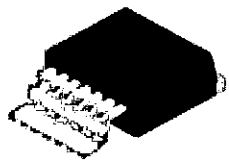
BTS 7960P

Product Summary

The **BTS 7960** is a fully integrated high current half bridge for motor drive applications. It is part of the **NovalithIC™** family containing one p-channel highside MOSFET and one n-channel lowside MOSFET with an integrated driver IC in one package. Due to the p-channel highside switch the need for a charge pump is eliminated thus minimizing EMI. Interfacing to a microcontroller is made easy by the integrated driver IC which features logic level inputs, diagnosis with current sense, slew rate adjustment, dead time generation and protection against overtemperature, overvoltage, undervoltage, overcurrent and short circuit.

The **BTS 7960** provides a cost optimized solution for protected high current PWM motor drives with very low board space consumption.

BTS 7960B
P-TO-263-7



BTS 7960P
P-TO-220-7

Basic Features

- Path resistance of typ. 16 mΩ @ 25 °C
- Low quiescent current of typ. 7 µA @ 25 °C
- PWM capability of up to 25 kHz combined with active freewheeling
- Switched mode current limitation for reduced power dissipation in overcurrent
- Current limitation level of 43 A typ.
- Status flag diagnosis with current sense capability
- Overtemperature shut down with latch behaviour
- Overvoltage lock out
- Undervoltage shut down
- Driver circuit with logic level inputs
- Adjustable slew rates for optimized EMI

Type	Ordering Code	Package
BTS 7960B	Q67060-S6160	P-TO-263-7
BTS 7960P	on request	P-TO-220-7

1 Overview

The **BTS 7960** is part of the **NovalithIC™** family containing three separate chips in one package: One p-channel highside MOSFET and one n-channel lowside MOSFET together with a driver IC, forming a fully integrated high current half-bridge. All three chips are mounted on one common leadframe, using the chip on chip and chip by chip technology. The power switches utilize vertical MOS technologies to ensure optimum on state resistance. Due to the p-channel highside switch the need for a charge pump is eliminated thus minimizing EMI. Interfacing to a microcontroller is made easy by the integrated driver IC which features logic level inputs, diagnosis with current sense, slew rate adjustment, dead time generation and protection against overtemperature, overvoltage, undervoltage, overcurrent and short circuit. The BTS 7960 can be combined with other BTS 7960 to form H-bridge and 3-phase drive configurations.

1.1 Block Diagram

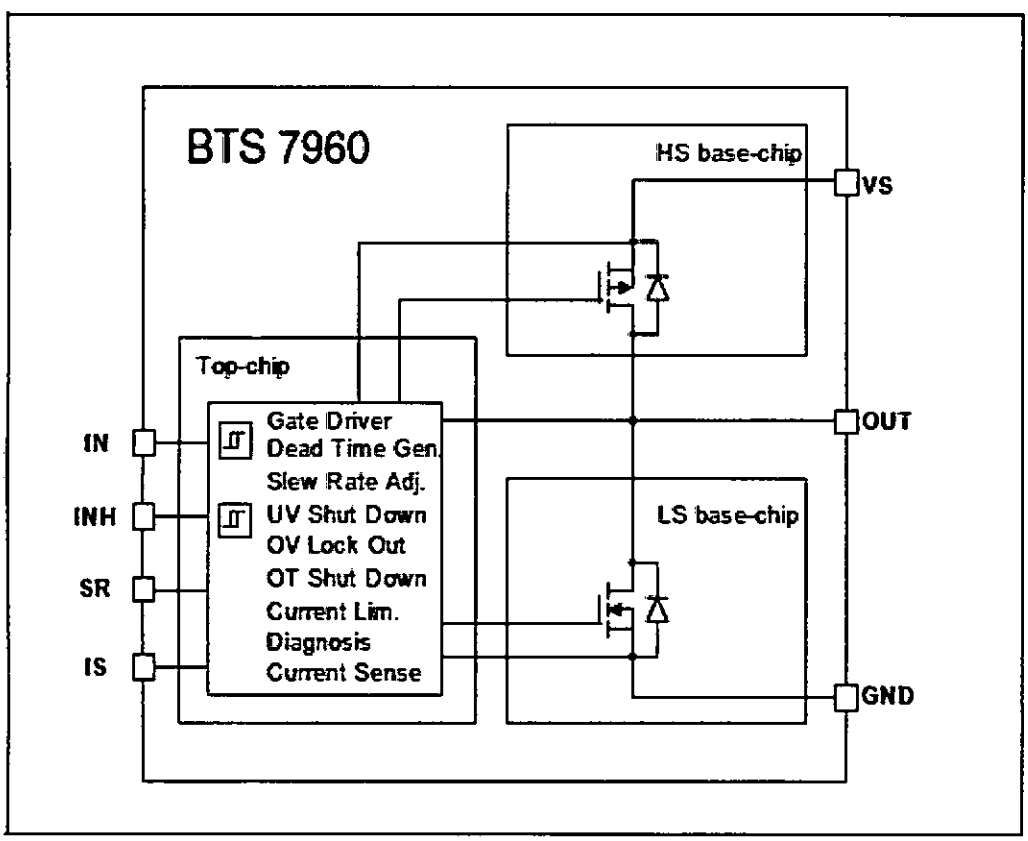


Figure 1 Block Diagram

1.2 Terms

Following figure shows the terms used in this data sheet.

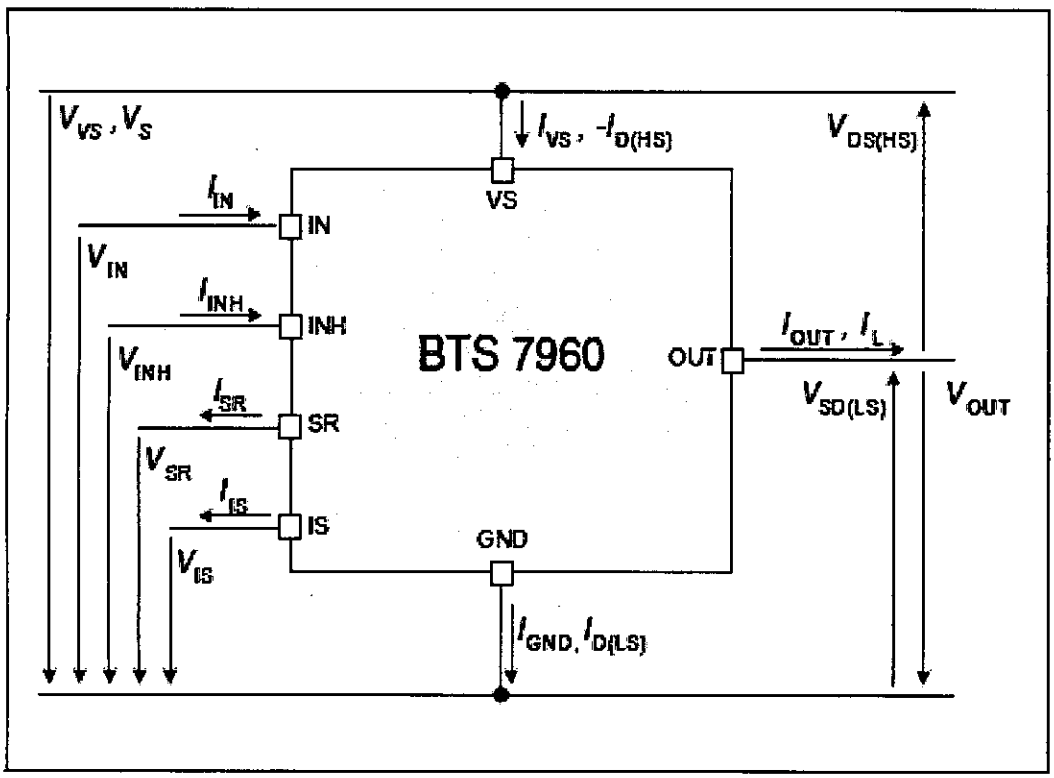


Figure 2 Terms

2 Pin Configuration

2.1 Pin Assignment

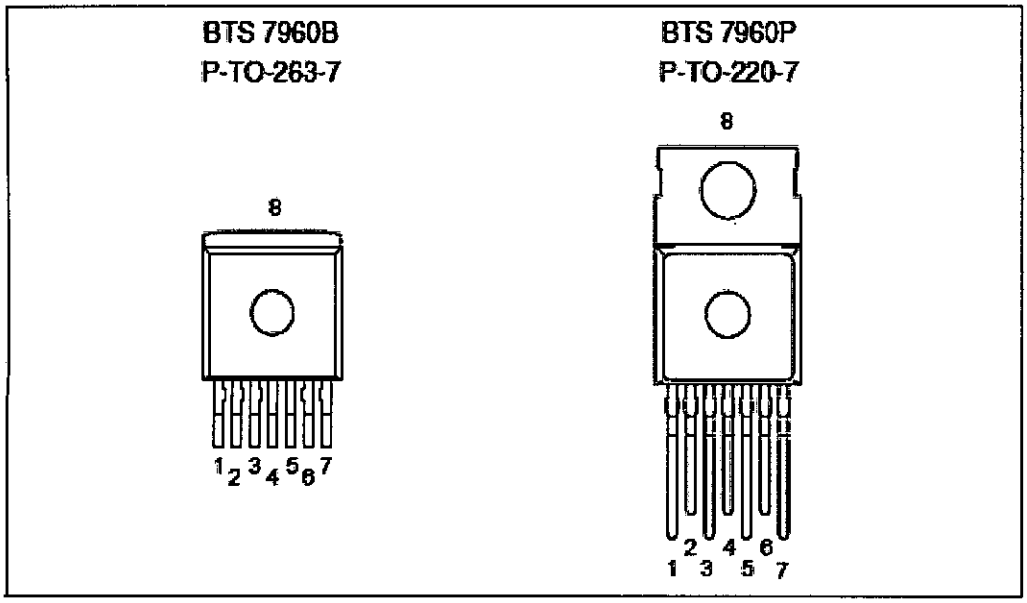


Figure 3 Pin Assignment BTS 7960B and BTS 7960P (top view)

2.2 Pin Definitions and Functions

Pin	Symbol	I/O	Function
1	**GND**	-	**Ground**
2	IN	I	Input Defines whether high- or lowside switch is activated
3	INH	I	Inhibit When set to low device goes in sleep mode
4,8	**OUT**	**O**	**Power output of the bridge**
5	SR	I	Slew Rate The slew rate of the power switches can be adjusted by connecting a resistor between SR and GND
6	IS	O	Current Sense and Diagnosis
7	**VS**	-	**Supply**

Bold type: pin needs power wiring

3 Maximum Ratings

-40 °C < T_j < 150 °C (unless otherwise specified)

Pos	Parameter	Symbol	Limits		Unit	Test Condition
			min	max		
Electrical Maximum Ratings						
3.0.1	Supply voltage	V_{VS}	-0.3	45	V	
3.0.2	Logic Input Voltage	V_{IN} V_{INH}	-0.3	5.3	V	
3.0.3	HS/LS continuous drain current	$I_{D(HS)}$ $I_{D(LS)}$	-40	40[1)]	A	T_C < 85°C switch active
3.0.4	HS pulsed drain current	$I_{D(HS)}$	-60	60[1)]	A	T_C < 85°C t_{pulse} = 10ms
3.0.5	LS pulsed drain current	$I_{D(LS)}$	-60	60[1)]	A	
3.0.6	Voltage at SR pin	V_{SR}	-0.3	1.0	V	
3.0.7	Voltage between VS and IS pin	V_{VS} - V_{IS}	-0.3	45	V	
3.0.8	Voltage at IS pin	V_{IS}	-20	45	V	
Thermal Maximum Ratings						
3.0.9	Junction temperature	T_j	-40	150	°C	
3.0.10	Storage temperature	T_{stg}	-55	150	°C	
ESD Susceptibility						
3.0.11	ESD susceptibility HBM	V_{ESD}			kV	according to EIA/ JESD 22-A 114B
	IN, INH, SR, IS		-2	2		
	OUT, GND, VS		-6	6		

1) Maximum reachable current may be smaller depending on current limitation level

Note: Maximum ratings are absolute ratings; exceeding any one of these values may cause irreversible damage to the device. Exposure to maximum rating conditions for extended periods of time may affect device reliability

4 Block Description and Characteristics

4.1 Supply Characteristics

$-40\,°C < T_j < 150\,°C$, $8\,V < V_S < 18\,V$, $I_L = 0A$ (unless otherwise specified)

Pos.	Parameter	Symbol	Limit Values			Unit	Test Conditions
			min.	typ.	max.		
General							
4.1.1	Operating Voltage	V_S	5.5	–	27.5	V	
4.1.2	Supply Current	$I_{VS(on)}$	–	2	3	mA	$V_{INH} = 5\,V$ $V_{IN} = 0\,V$ or $5\,V$ R_{SR}=0 Ω DC-mode normal operation (no fault condition)
4.1.3	Quiescent Current	$I_{VS(off)}$	–	7	15	µA	$V_{INH} = 0\,V$ $V_{IN} = 0\,V$ or $5\,V$ T_j <85 °C
			–	–	65	µA	$V_{INH} = 0\,V$ $V_{IN} = 0\,V$ or $5\,V$

HC Serial Bluetooth Products

User Instructional Manual

1 Introduction

HC serial Bluetooth products consist of Bluetooth serial interface module and Bluetooth adapter, such as:

(1) Bluetooth serial interface module:

Industrial level: HC-03, HC-04(HC-04-M, HC-04-S)

Civil level: HC-05, HC-06(HC-06-M, HC-06-S)

HC-05-D, HC-06-D (with baseboard, for test and evaluation)

(2) Bluetooth adapter:

HC-M4

HC-M6

This document mainly introduces Bluetooth serial module. Bluetooth serial module is used for converting serial port to Bluetooth. These modules have two modes: master and slaver device. The device named after even number is defined to be master or slaver when out of factory and can't be changed to the other mode. But for the device named after odd number, users can set the work mode (master or slaver) of the device by AT commands.

HC-04 specifically includes:

Master device: HC-04-M, M=master

Slave device: HC-04-S, S=slaver

The default situation of HC-04 is slave mode. If you need master mode, please state it clearly or place an order for HC-04-M directly.The naming rule of HC-06 is same.

When HC-03 and HC-05 are out of factory, one part of parameters are set for activating the device. The work mode is not set, since user can set the mode of HC-03, HC-05 as they want.

The main function of Bluetooth serial module is replacing the serial port line, such as:

1. There are two MCUs want to communicate with each other. One connects to Bluetooth master device while the other one connects to slave device. Their connection can be built once the pair is made. This Bluetooth connection is equivalently liked to a serial port line connection including RXD, TXD

signals. And they can use the Bluetooth serial module to communicate with each other.

2. When MCU has Bluetooth salve module, it can communicate with Bluetooth adapter of computers and smart phones. Then there is a virtual communicable serial port line between MCU and computer or smart phone.

3. The Bluetooth devices in the market mostly are salve devices, such as Bluetooth printer, Bluetooth GPS. So, we can use master module to make pair and communicate with them.

Bluetooth Serial module's operation doesn't need drive, and can communicate with the other Bluetooth device who has the serial. But communication between two Bluetooth modules requires at least two conditions:

(1) The communication must be between master and slave.

(2) The password must be correct.

However, the two conditions are not sufficient conditions. There are also some other conditions basing on different device model. Detailed information is provided in the following chapters.

In the following chapters, we will repeatedly refer to Linvor's (Formerly known as Guangzhou HC Information Technology Co., Ltd.) material and photos.

2 Selection of the Module

The Bluetooth serial module named even number is compatible with each other, The salve module is also compatible with each other. In other word, the function of HC-04 and HC-06, HC-03 and HC-05 are mutually compatible with each other. HC-04 and HC-06 are former version that user can't reset the work mode (master or slave). And only a few AT commands and functions can be used, like reset the name of Bluetooth (only the slaver), reset the password, reset the baud rate and check the version number. The command set of HC-03 and HC-05 are more flexible than HC-04 and HC-06's. Generally, the Bluetooth of HC-03/HC-05 is recommended for the user.

Here are the main factory parameters of HC-05 and HC-06. Pay attention to the differences:

HC-05	HC-06
Master and slave mode can be switched	Master and slave mode can't be switched
Bluetooth name: HC-05	Bluetooth name: linvor
Password:1234	Password:1234

Master role: have no function to remember the last paired salve device. It can be made paired to any slave device. In other words, just set AT+CMODE=1 when out of factory. If you want HC-05 to remember the last paired slave device address like HC-06, you can set AT+CMODE=0 after paired with the other device. Please refer the command set of HC-05 for the details.	Master role: have paired memory to remember last slave device and only make pair with that device unless KEY (PIN26) is triggered by high level. The default connected PIN26 is low level.
Pairing: The master device can not only make pair with the specified Bluetooth address, like cell-phone, computer adapter, slave device, but also can search and make pair with the slave device automatically. Typical method: On some specific conditions, master device and slave device can make pair with each other automatically. (This is the default method.)	Pairing: Master device search and make pair with the slave device automatically. Typical method: On some specific conditions, master and slave device can make pair with each other automatically.
Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.	Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.
AT Mode 1: After power on, it can enter the AT mode by triggering PIN34 with high level. Then the baud rate for setting AT command is equal to the baud rate in communication, for example: 9600. AT mode 2: First set the PIN34 as high level, or while on powering the module set the PIN34 to be high level, the Baud rate used here is 38400 bps. Notice: All AT commands can be operated only	AT Mode: Before paired, it is at the AT mode. After peired it's at transparent communication.

when the PIN34 is at high level. Only part of the AT commands can be used if PIN34 doesn't keep the high level after entering to the AT mode. Through this kind of designing, set permissions for the module is left to the user's external control circuit, that makes the application of HC-05 is very flexible.	
During the process of communication, the module can enter to AT mode by setting PIN34 to be high level. By releasing PIN34, the module can go back to communication mode in which user can inquire some information dynamically. For example, to inquire the pairing is finished or not.	During the communication mode, the module can't enter to the AT mode.
Default communication band rate: 9600, 4800-1.3M are settable.	Default communication band rate: 9600, 1200-1.3M are settable.
KEY: PIN34, for entering to the AT mode.	KEY: PIN26, for master abandons memory.
LED1: PIN31, indicator of Bluetooth mode. Slow flicker (1Hz) represents entering to the AT mode2, while fast flicker(2Hz) represents entering to the AT mode1 or during the communication pairing. Double flicker per second represents pairing is finished, the module is communicable. LED2: PIN32, before pairing is at low level, after the pairing is at high level. The using method of master and slaver's indicator is the same. Notice: The PIN of LED1 and LED2 are connected with LED+.	LED: The flicker frequency of slave device is 102ms. If master device already has the memory of slave device, the flicker frequency during the pairing is 110ms/s. If not, or master has emptied the memory, then the flicker frequency is 750m/s. After pairing, no matter it's a master or slave device, the LED PIN is at high level. Notice: The LED PIN connects to LED+ PIN.
Consumption: During the pairing, the current is	Consumption: During the pairing, the current is

fluctuant in the range of 30-40mA. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.	fluctuant in the range of 30-40 m. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.
Reset: PIN11, active if it's input low level. It can be suspended in using.	Reset: PIN11, active if it's input low level. It can be suspended in using.
Level: Civil	Level: Civil

The table above that includes main parameters of two serial modules is a reference for user selection.

HC-03/HC-05 serial product is recommended.

3. Information of Package

The PIN definitions of HC-03, HC-04, HC-05 and HC-06 are kind of different, but the package size is the same: 28mm * 15mm * 2.35mm.

The following figure 1 is a picture of HC-06 and its main PINs. Figure 2 is a picture of HC-05 and its main PINs. Figure 3 is a comparative picture with one coin. Figure 4 is their package size information. When user designs the circuit, you can visit the website of Guangzhou HC Information Technology Co., Ltd. (www.wavesen.com) to download the package library of protle version.

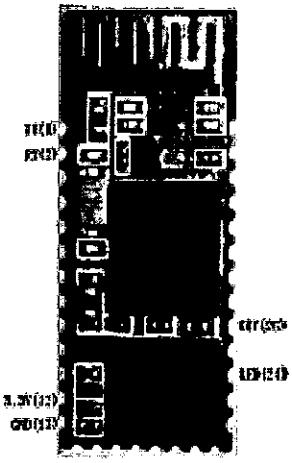


Figure 1 HC-06

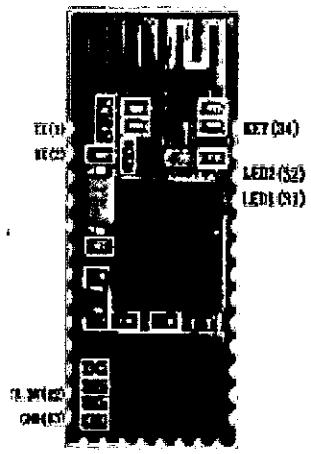
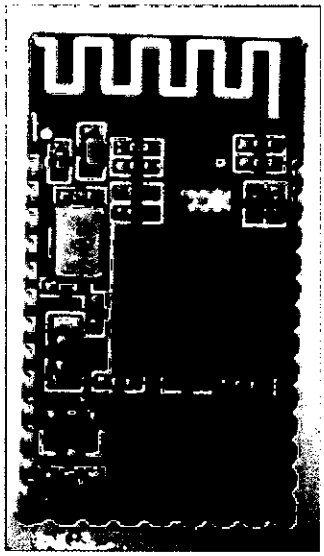
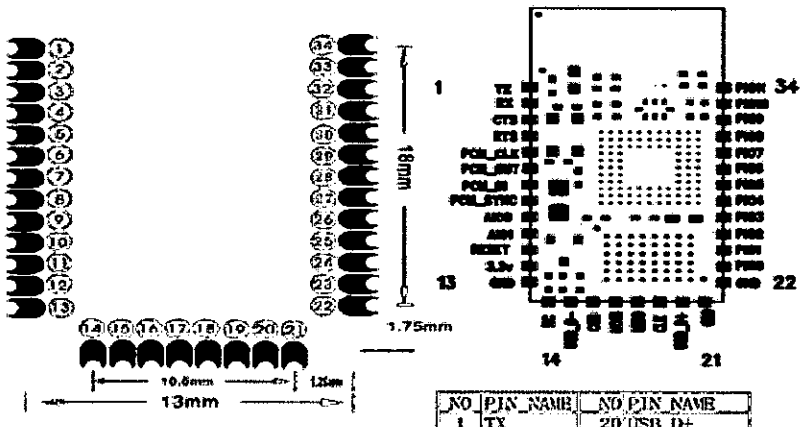
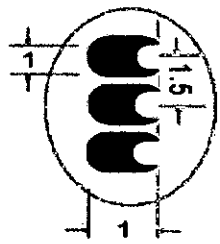


Figure 2 HC-05





NO	PIN_NAME	NO	PIN_NAME
1	TX	20	USB_D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM_CLK	24	PIO1
6	PCM_OUT	25	PIO2
7	PCM_IN	26	PIO3
8	PCM_SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB_D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		



PCB Layout 请参考实物

Figure 4 Package size information

LINVOR BLUE T

www.linvor.com

LV-BC-2.0

单位： mm

4. The Using and Testing Method of HC-06 for the First Time

This chapter will introduce the using method of HC-06 in detail. User can test the module according to this chapter when he or she uses the module at the first time.

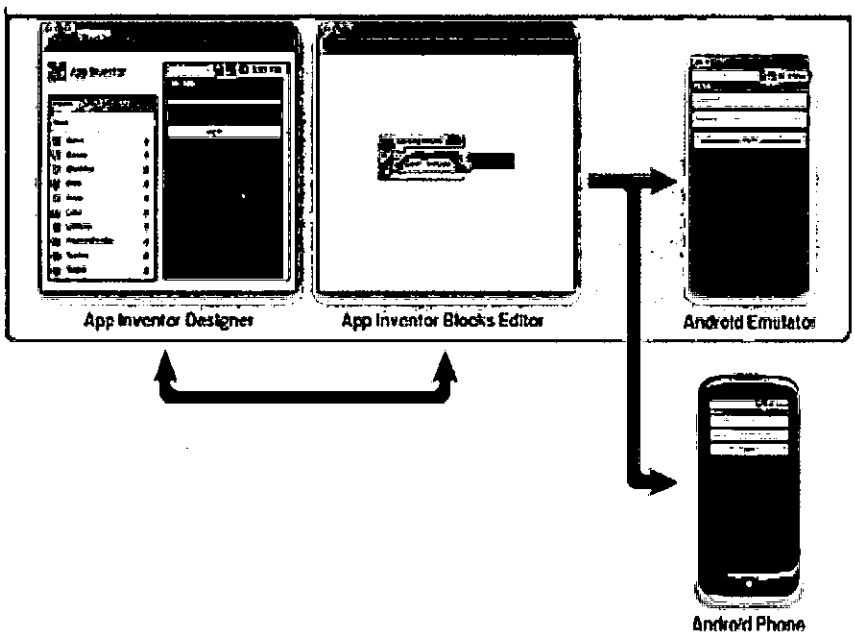
PINs description:

PIN1	UART_TXD , TTL/CMOS level, UART Data output
PIN2	UART_RXD, TTL/COMS level, s UART Data input
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND
PIN22	GND
PIN24	LED, working mode indicator Slave device: Before paired, this PIN outputs the period of 102ms square wave. After paired, this PIN outputs high level. Master device: On the condition of having no memory of pairing with a slave device, this PIN outputs the period of 110ms square wave. On the condition of having the memory of pairing with a slave device, this PIN outputs the period of 750ms square wave. After paired, this PIN outputs high level.
PIN26	For master device, this PIN is used for emptying information about pairing. After emptying, master device will search slaver randomly, then remember the address of the new got slave device. In the next power on, master device will only search this address.

MIT App Inventor Getting Started Guide

What is App Inventor?

App Inventor lets you develop applications for Android phones using a web browser and either a connected phone or an on-screen phone emulator. The MIT App Inventor servers store your work and help you keep track of your projects.



You build apps by working with:

- The *App Inventor Designer*, where you select the components for your app.
- The *App Inventor Blocks Editor*, where you assemble program blocks that specify how the components should behave. You assemble programs visually, fitting pieces together like pieces of a puzzle.

Your app appears on the phone step-by-step as you add pieces to it, so you can test your work as you build. If you don't have an Android phone, you can build your apps using the *Android emulator*, software that runs on your computer and behaves just like the phone.

The App Inventor development environment is supported for Mac OS X, GNU/Linux, and Windows operating systems, and several popular Android phone models. Applications created with App Inventor can be installed on any Android phone.

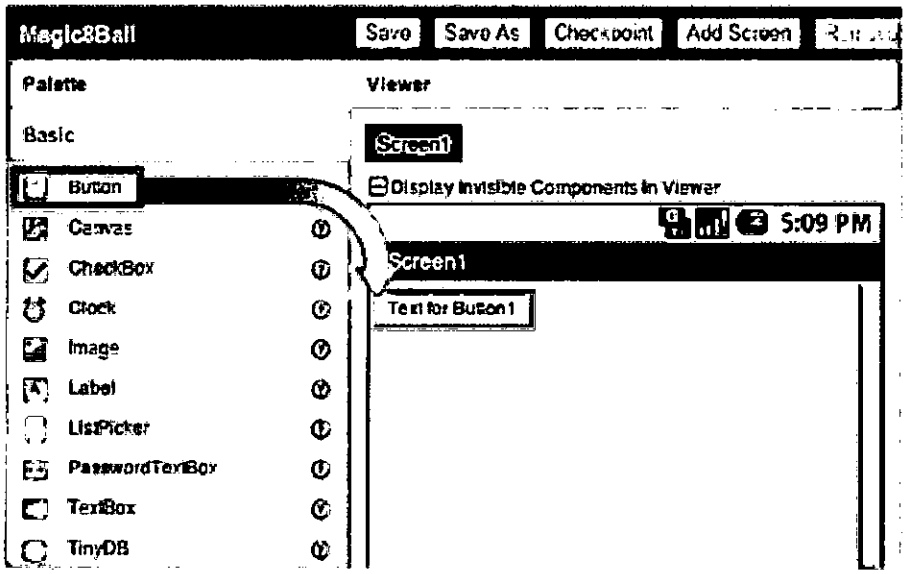
Before you can use App Inventor, you need to set up your computer and install the *App Inventor Setup* package on your computer. See: http://explore.appinventor.mit.edu/content/setup

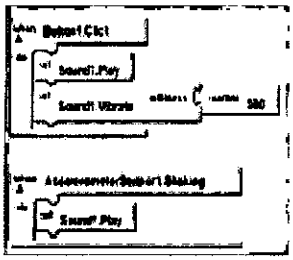
What can I do with App Inventor?

You can build many different types of apps with App Inventor. Often people begin by building games like MoleMash or games that let you draw funny pictures on your friend's faces. You can even make use of the phone's sensors to move a ball through a maze based on tilting the phone.

But app building is not limited to simple games. You can also build apps that inform and educate. You can create a quiz app to help you and your classmates study for a test. With Android's text-to-speech capabilities, you can even have the phone ask the questions aloud.

To use App Inventor, you do not need to be a professional developer. This is because instead of writing code, you visually design the way the app looks and use blocks to specify the app's behavior.





The App Inventor team has created blocks for just about everything you can do with an Android phone, as well as blocks for doing "programming-like" stuff– blocks to store information, blocks for repeating actions, and blocks to perform actions under certain conditions. There are even blocks to talk to services like Twitter.

Simple but Powerful!

App Inventor is simple to use, but also very powerful. Apps you build can even store data created by users in a database, so you can create a make-a-quiz app in which the teachers can save questions in a quiz for their students to answer.

Because App Inventor provides access to a GPS-location sensor, you can build apps that know where you are. You can build an app to help you remember where you parked your car, an app that shows the location of your friends or colleagues at a concert or conference, or your own custom tour app of your school, workplace, or a museum.

You can write apps that use the phone features of an Android phone. You can write an app that periodically texts "missing you" to your loved ones, or an app "No Text While Driving" that responds to all texts automatically with "sorry, I'm driving and will contact you later". You can even have the app read the incoming texts aloud to you (though this might lure you into responding).

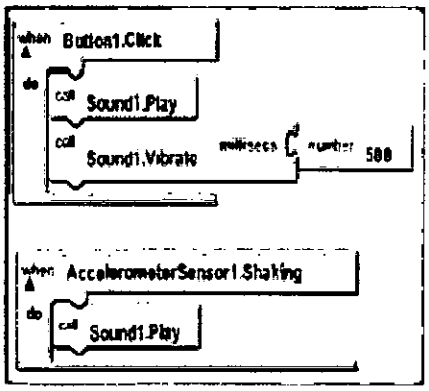
App Inventor provides a way for you to communicate with the web. If you know how to write web apps, you can use App Inventor to write Android apps that talk to your favorite web sites, such as Amazon and Twitter.

Understanding App Inventor Programming

Event Handlers

App Inventor programs describe how the phone should respond to certain events: a button has been pressed, the phone is being shaken, the user is dragging her finger over a canvas, etc. This is specified by event handler blocks, which used the word when. E.g., when Button1.Click and when AccelerometerSensor1.Shaking in HelloPurr.

Most event handlers are in green color and stored at the top part of each drawer. Here are the example of event handlers.

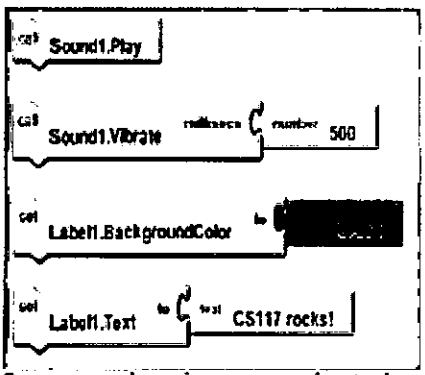


When an event occurs on a phone, the corresponding event handler is said to fire, which means it is executed.

Commands and Expressions

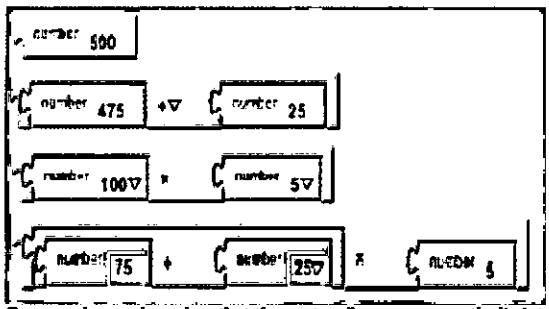
When an event handler fires, it executes a sequence of commands in its body. A command is a block that specifies an action to be performed on the phone (e.g., playing sounds). Most command blocks are in purple or blue color.

Here are some sample commands available in HelloPurr:

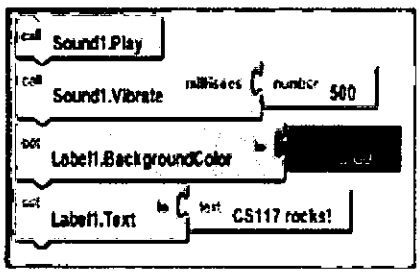


Some commands require one or more input values (also known as parameters or arguments) to completely specify their action. For example, call Sound1.Vibrate needs to know the number of milliseconds to vibrate, set Label1.BackgroundColor needs to know the new background color of the label, and set Label1.text needs to know the new text string for the label. The need for input values is shown by sockets on the right edge of the command.

These sockets can be filled with expressions, blocks that denote a value. Expression blocks have leftward-pointing plugs that you can imagine transmit the value to the socket. Larger expressions can be built out of simpler ones by horizontal composition. E.g., all of the following expressions denote the number 500:



Commands are shaped so that they naturally compose vertically into a command stack, which is just one big command built out of smaller ones. Here's a stack with four commands:



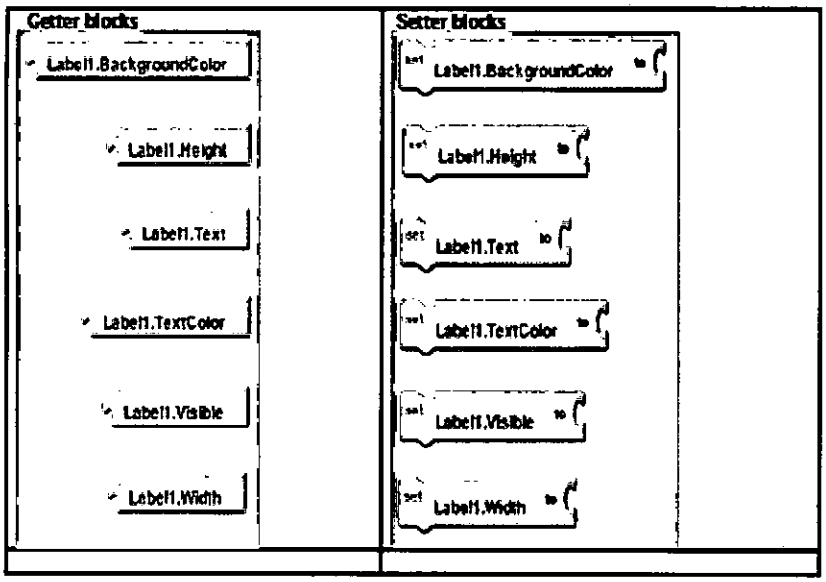
When this stack of commands are placed in a body of an event handler (e.g., the *when.Button1.Click* event handler), the command will be executed from the top to the bottom. If the stack of command above is executed, then the phone will first play the sound, then vibrate, then change the label's color to be green, and then label will show the text "CS117 rocks!"

However, the execution works very fast: you would see all the actions happen at the same time.

Control Flow

When an event handler fires, you can imagine that it creates a karaoke-like control dot that flows through the command stack in its body. The control dot moves from the top of the stack to the bottom, and when it reaches a command, that command is executed – i.e, the action of that command is performed. Thinking about control "flowing" through a program will help us understand its behavior.

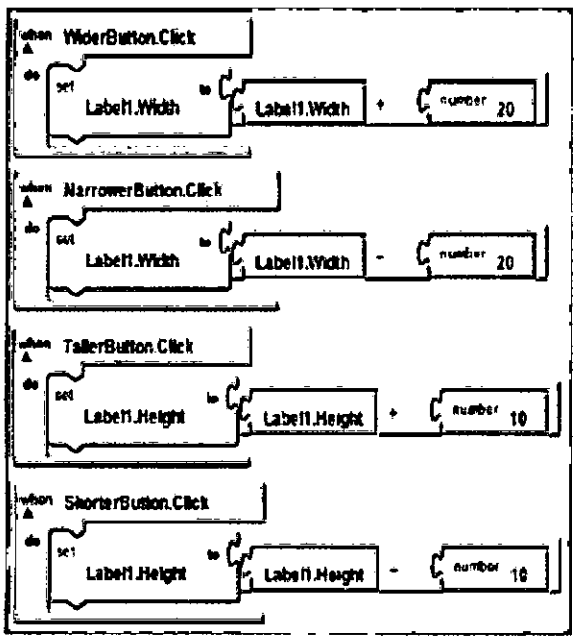
The order of the commands, or the *control flow* is important when you make an app. You need to make sure which action should come first.



Getter blocks are expressions that get the current value of the property. Setter blocks are commands that change the value associated with the property.

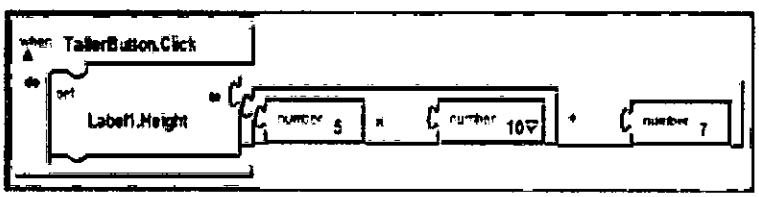
Some Label properties cannot be manipulated by blocks. Which ones?

As an example of manipulating Label properties, open the LabelSize program, which has 4 event handlers. What do these do?

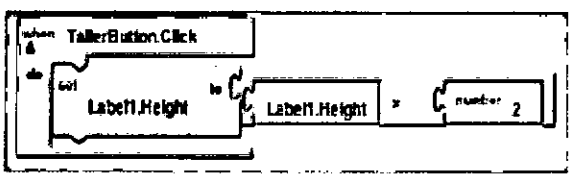


Predict what happens if we change the when TallerButton.Click handler as follows:

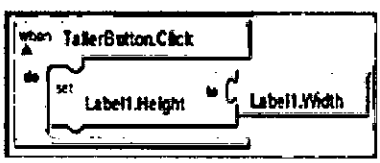
Modification 1:



Modification 2:

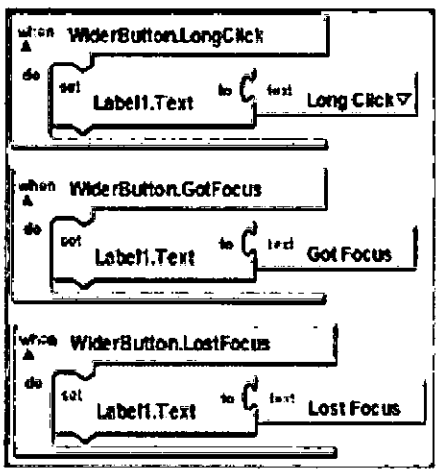


Modification 3:



Other Button Events

Other button events are when LongClick, when GotFocus, and when LostFocus. Experiment with these by creating the following handlers:



Note: many components get focus when touched, and lose it when the touch is removed. But buttons are special, because touching them fires the Click event. However, they can get/lose focus through the G1 track ball or IDEOS navigator.

Renaming Components

Programs can be easier to read if you change the default name of components.
E.g., NarrowerButton is more meaningful than Button2. In the Components pane of the Designer window, use the Rename button to rename Label1 to MyLabel. What happens in the Blocks Editor to blocks that used to mention Label1?

The TextBox Component

Many apps expect users to enter input like numbers or text strings. The TextBox component is used for this purpose.

Let's add two text boxes to the LabelSize program:

- The first one should specify the amount by which the label should become wider/narrower.
- The second one should specify a string to be concatenated with the current label. (Use the join operator for this.)

Note: Sometimes it is useful to have text strings with multiple lines. In a string, the notation \n stands for the newline character. For example, the text string one\ntwo\nthree has three lines.

The Canvas Component

In PaintPot, you met a new component, Canvas, that is used for drawing and animation.
Upload CanvasTest2.zip (attached at the bottom of this page: https://sites.google.com/site/wellesleycs117fall11/lecture-notes/lecture-04-animation-components) to create the CanvasTest2 app.

ANEXOS

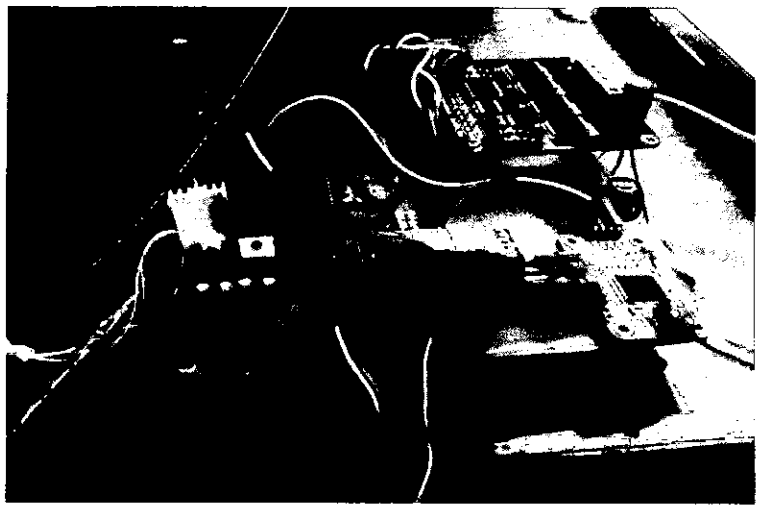


Foto 1.1 Tarjeta Controladora de la Silla de Ruedas.

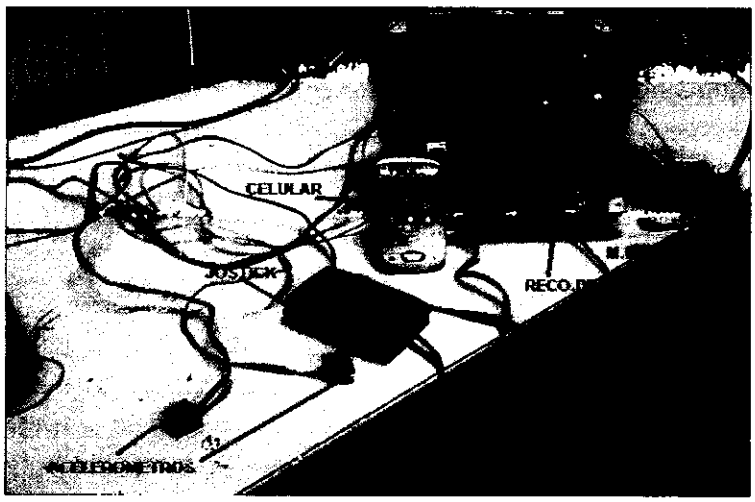


Foto 1.2 Componentes del Sistema de Control de la silla de ruedas

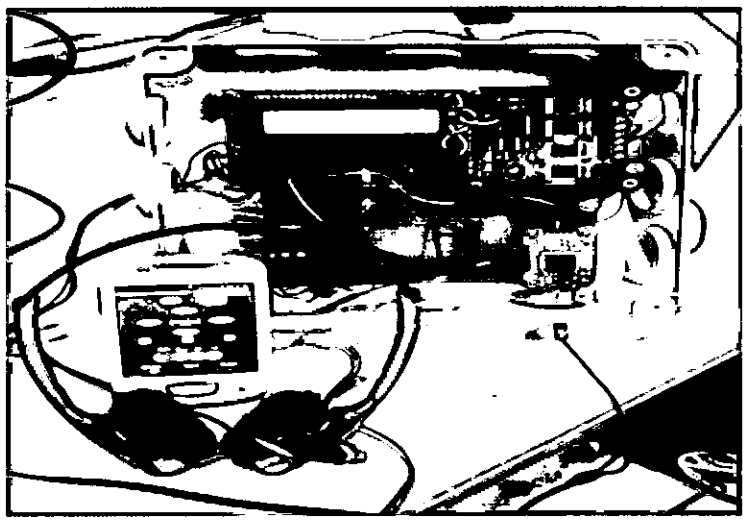


Foto 1.3 Componentes del sistema de Control de la silla de ruedas

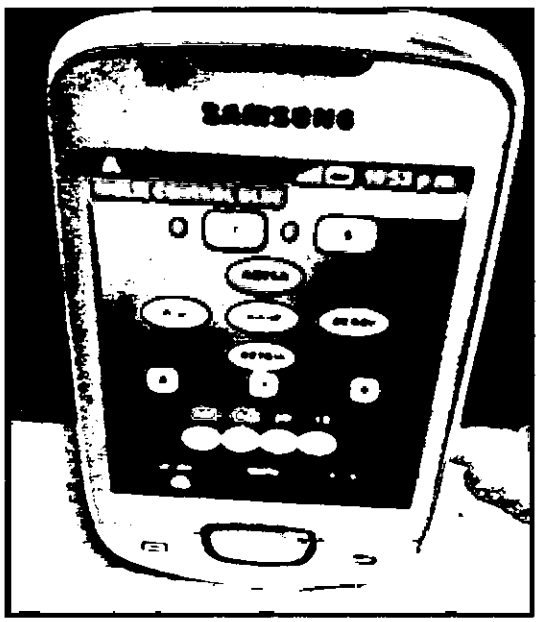


Foto1.4 Aplicación Instalada en Celular

Adaptación de Motores a una Silla de ruedas Común

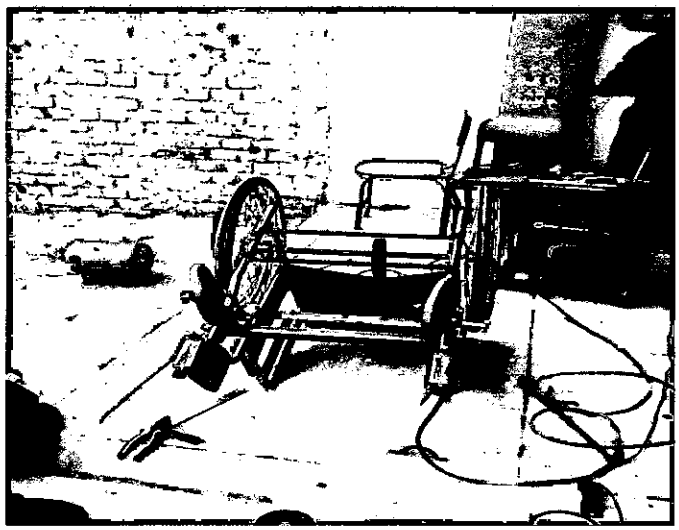


Foto1.5 Acondicionamiento de la silla

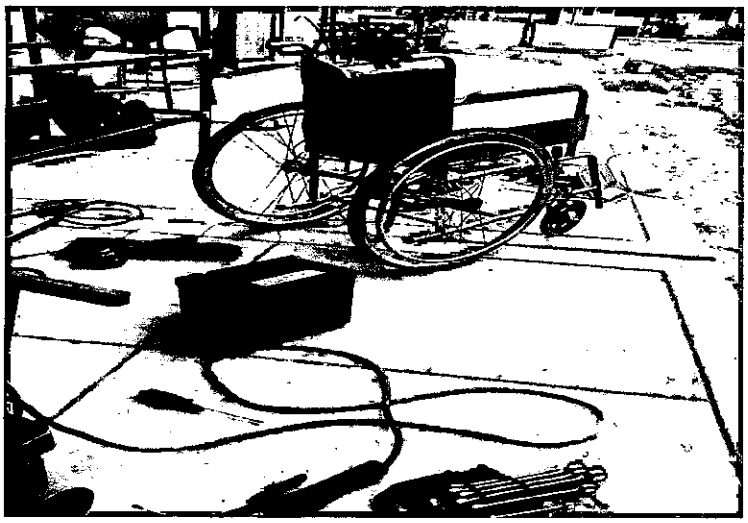


Foto 1.6 Acondicionamiento de la silla

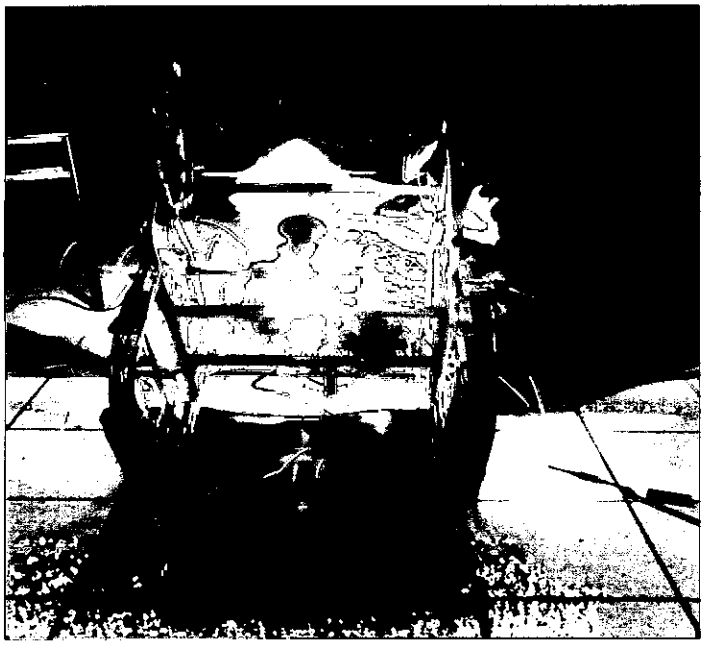


Foto 1.7 Acondicionamiento de la silla

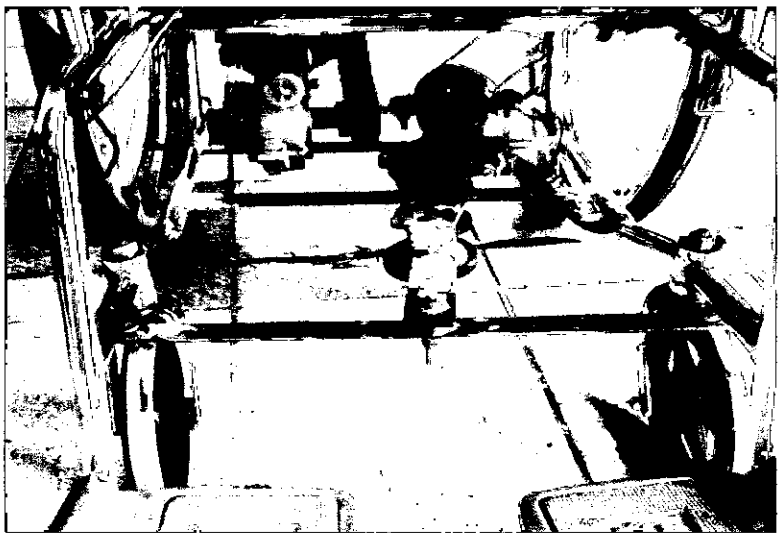


Foto1.8 Acondicionamiento de la silla, instalando motores

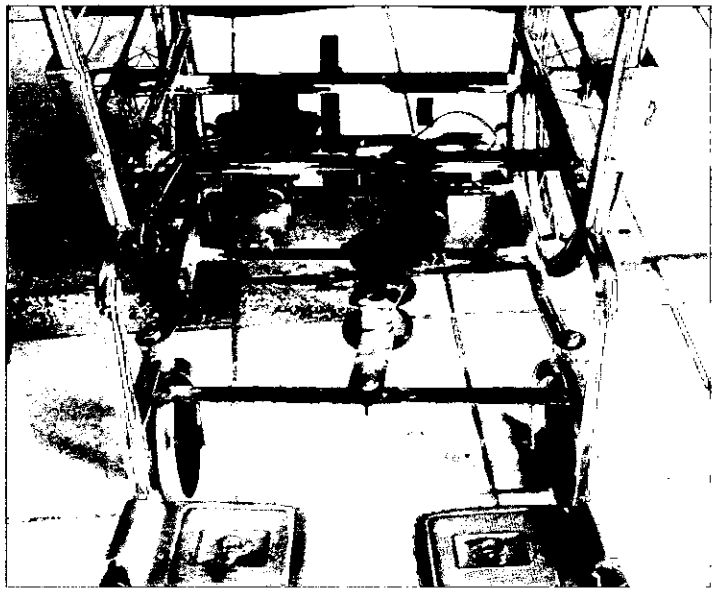


Foto 1.9 Acondicionamiento de la silla, instalando motores

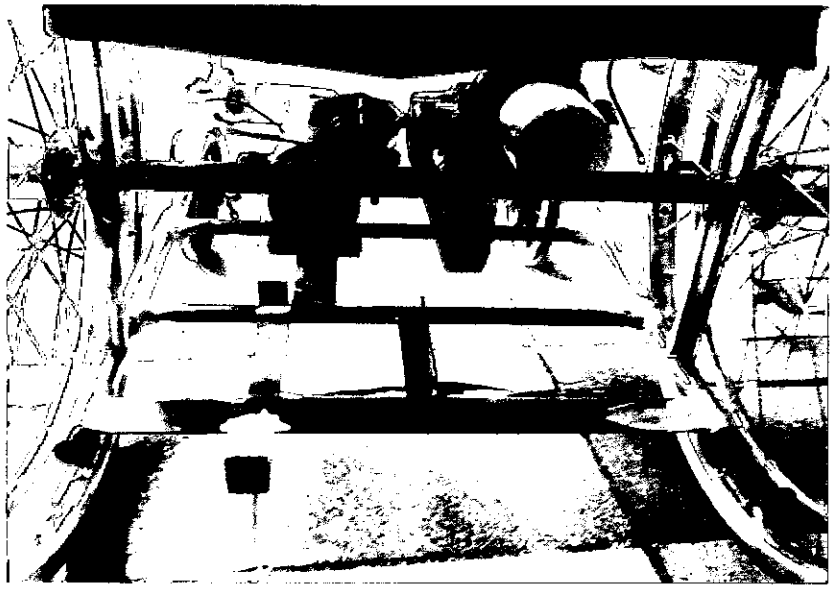


Foto1.10 Acondicionamiento de la silla, instalando motores

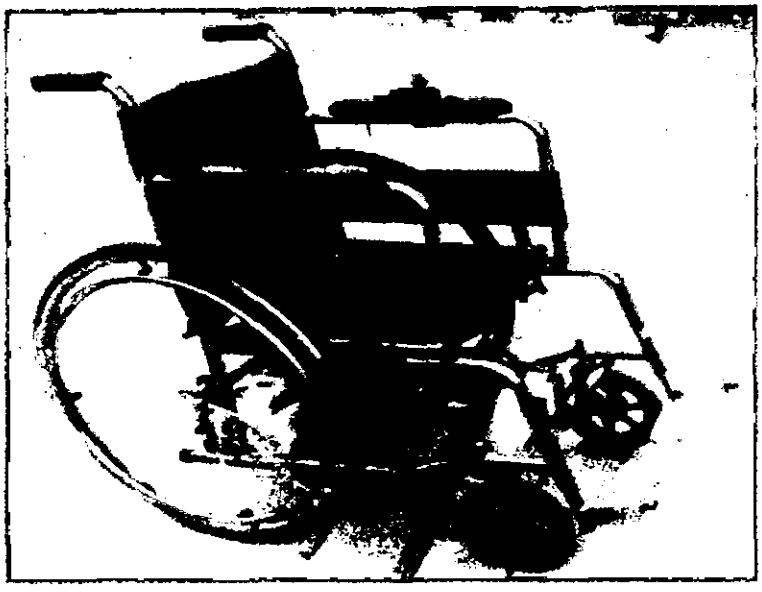


Foto1.11 Silla de Ruedas culminada

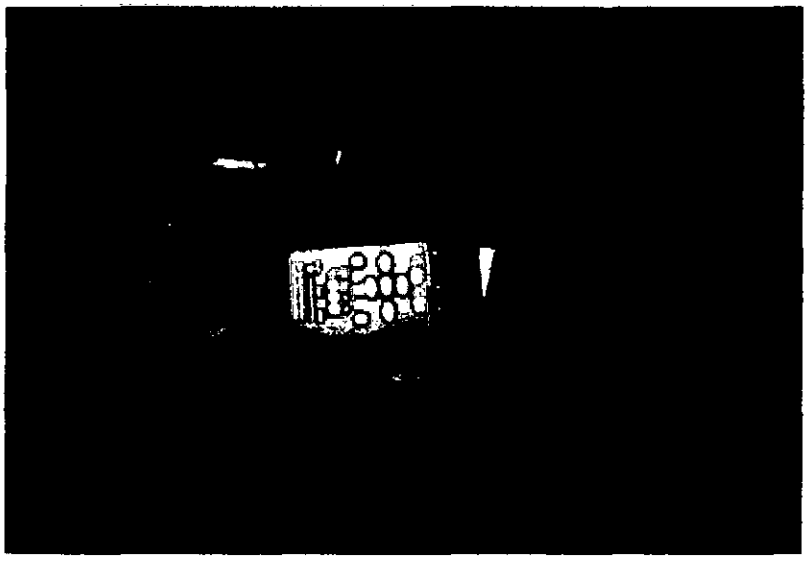


Foto1.12 Probando la silla de ruedas y la comunicación con celular

Foto 1.13 Probando la silla de ruedas y la comunicación con celular